UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

**CENTRO DE CIENCIAS BÁSICAS**

**DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN**

**TESIS**

# "DISEÑO Y USO DE PLANES DE PROGRAMACIÓN COMO APOYO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN"

**PRESENTA**

**Blanca Guadalupe Estrada Renteria**

**PARA OBTENER EL GRADO DE MAESTRA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES**

**TUTOR**

Tutor: Dr. Carlos A. Arévalo Mercado

**INTEGRANTES DEL COMITÉ TUTORAL**

Dra. Estela Lizbeth Muñoz Andrade

Dra. Lizeth Itziguery Solano Romo

Aguascalientes Ags., Mayo de 2018

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

**BLANCA GUADALUPE ESTRADA RENTERIA**
**MAESTRÍA EN INFORMÁTICA Y TECNOLOGÍAS COMPUTACIONALES**
**P R E S E N T E.**

Estimada alumna:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o caso práctico titulado: **"DISEÑO Y USO DE PLANES DE PROGRAMACIÓN COMO APOYO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN",** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

**A T E N T A M E N T E**
Aguascalientes, Ags., a 22 de mayo de 2018
*"Se lumen proferre"*
**EL DECANO**

**M. en C. JOSÉ DE JESÚS RUÍZ GALLEGOS**

c.c.p.- Archivo.

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutor designado del estudiante **BLANCA GUADALUPE ESTRADA RENTERIA** con ID 7392 quien realizó la tesis titulada: **DISEÑO Y USO DE PLANES DE PROGRAMACIÓN COMO APOYO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN,** y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que pueda proceder a imprimirla y así como continuar con el procedimiento administrativo para la obtención del grado.

Someto lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
*"Se Lumen Proferre"*
Aguascalientes, Ags., a 22 de mayo de 2018.

**DR. CARLOS ARGELIO ARÉVALO MERCADO**
TUTOR DE TESIS
PROFESOR INVESTIGADOR,
DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN
CENTRO DE CIENCIAS BÁSICAS

c.c.p.- Interesado
c.c.p.- Dr. José Manuel Mora Tavarez. Secretario Técnica de la Maestría en Informática y Tecnologías Computacionales.
c.c.p.- Dr. Rogelio Salinas, Secretario de investigación y Posgrado del Centro de Ciencias Básicas.
c.c.p.- Dr. César Eduardo Velazquez Amador, Consejero Técnico de la Maestría en Informática y Tecnologías Computacionales.

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutora designada de la estudiante **BLANCA GUADALUPE ESTRADA RENTERIA** con ID 7392 quien realizó la tesis titulada: **DISEÑO Y USO DE PLANES DE PROGRAMACIÓN COMO APOYO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN,** y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que pueda proceder a imprimirla y así como continuar con el procedimiento administrativo para la obtención del grado.

Someto lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
*"Se Lumen Proferre"*
Aguascalientes, Ags., a 22 de mayo de 2018.

**DRA. ESTELA LIZBETH MUÑOZ ANDRADE**
PROFESOR - INVESTIGADOR
DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS
CENTRO DE CIENCIAS BÁSICAS

c.c.p.- Interesado
c.c.p.- Dr. José Manuel Mora Tavarez. Secretario Técnica de la Maestría en Informática y Tecnologías Computacionales.
c.c.p.- Dr. Rogelio Salinas, Secretario de investigación y Posgrado del Centro de Ciencias Básicas.
c.c.p.- Dr. César Eduardo Velazquez Amador, Consejero Técnico de la Maestría en Informática y Tecnologías Computacionales.

UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. EN C. JOSE DE JESUS RUIZ GALLEGOS
DECANO DEL CENTRO DE CIENCIAS BASICAS
P R E S E N T E

Por medio del presente como Tutor designado de la estudiante **BLANCA GUADALUPE ESTRADA RENTERIA** con ID 7392 quien realizó la tesis titulada: **DISEÑO Y USO DE PLANES DE PROGRAMACIÓN COMO APOYO PARA LA ENSEÑANZA DE LA PROGRAMACIÓN,** y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que pueda proceder a imprimirla y así como continuar con el procedimiento administrativo para la obtención del grado.

Someto lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
*"Se Lumen Proferre"*
Aguascalientes, Ags., a 22 de mayo de 2018.

**DRA. LIZETH ITZIGUERY SOLANO ROMO**
TUTOR DE TESIS
PROFESOR INVESTIGADOR,
DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN
CENTRO DE CIENCIAS BÁSICAS

c.c.p.- Interesado
c.c.p.- Dr. José Manuel Mora Tavarez. Secretario Técnica de la Maestría en Informática y Tecnologías Computacionales.
c.c.p.- Dr. Rogelio Salinas, Secretario de investigación y Posgrado del Centro de Ciencias Básicas.
c.c.p.- Dr. César Eduardo Velazquez Amador, Consejero Técnico de la Maestría en Informática y Tecnologías Computacionales.

# AGRADECIMIENTOS

Agradezco desde lo más profundo de mi corazón:

Primeramente, a Dios, que me ha llenado de bendiciones a lo largo de mi vida y ahora me permite un logro académico más.

Después a Guillermo Domínguez Aguilar mi amor y compañero de vida, por ser parte fundamental en mi crecimiento académico, por ser mi soporte y estar siempre a mi lado y sobre todo por su amor y paciencia.

A mis dos amores más grandes que son mis hijas Estefanía y Arely, por ser mi motivo de vida y por soportar este tiempo dedicado a mis estudios sin protestar.

De una manera muy especial quiero agradecer al Dr. Carlos A. Arévalo Mercado, por ser primeramente una gran persona y un excelente tutor, por toda su dirección y apoyo para la elaboración de esta tesis ya que fue parte fundamental de la misma.

A la Dra. Lizbeth Muñoz Andrade por ser primeramente mi amiga, por estar siempre al pendiente de mis estudios, por impulsarme y por ayudarme, por el tiempo y el esfuerzo dedicado a apoyar este trabajo.

A la Dra. Lizeth Itziguery Solano Romo por su apoyo y a la Mtra. Georgina Salazar Partida por el apoyo brindado para la prueba de los materiales de esta tesis.

A cada uno de mis profesores por su dedicación y tiempo entregados.

Finalmente, a mi casa de estudios la Universidad Autónoma de Aguascalientes, que me ha dado tanto académicamente y profesionalmente.

**Se Lumen Proferre**

# DEDICATORIAS

**A Dios**

Por permitirme un logro más en mi vida

**A mis padres†**

Que me enseñaron a ser la persona que ahora soy

**A mi esposo y a mis hijas**

Por su amor, apoyo y soporte y por ser la

parte más fundamental de mi vida

**A mis familiares**

Que siempre me apoyan y cree en mi

**A mi comité tutoral**

Por el apoyo y guía para la realización de esta tesis

# INDICE GENERAL

# INDICE DE TABLAS

# INDICE DE ILUSTRACIONES

# RESUMEN ESPAÑOL

Tomando como referencia la carga cognitiva, este trabajo de investigación tiene como objetivo diseñar, desarrollar y probar material didáctico basado en identificación de planes de programación, esto con el propósito de medir su efectividad en el aprendizaje de la programación en alumnos de primer año del programa educativo de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes.

Las teorías base sobre las cuales se sustenta este estudio son: Teoría de la Carga Cognitiva, Problemas por completar y Planes de programación.

La teoría de la carga cognitiva se basa en dos ideas. La primera indica que la cantidad de información nueva que procesa el cerebro es limitada, mientras que la segunda indica que no hay límite en la cantidad de información procesada por el cerebro de la cual ya se tiene almacenada. La teoría de la carga cognitiva respalda los modelos de enseñanza explícitos, porque dichos modelos siguen los principios de cómo el cerebro aprende de forma más efectiva (Kirschner, Sweller & Clark 2006).

La teoría de los problemas por completar, Merriënboer & Krammer (1987) señala que los alumnos prestan una mayor atención a los ejemplos resueltos cuando se les proporciona a los alumnos ejemplos por completar. Un problema por completar es un problema parcialmente resuelto en el cual el estudiante tiene que completar la solución al mismo, es importante que cuando se elaboren este tipo de ejemplos por completar, se preste particular atención en que el estudiante complete aquellas partes que se pretende formen un “esquema” de solución en su memoria a largo plazo.

Y finalmente un plan de programación se define como una secuencia de instrucciones para solucionar problemas que se desarrollan de la misma manera siempre, es decir, las soluciones siguen un plan uniforme. Un caso típico de éstos son los acumuladores, mismos que se pueden usar en distintas y muy variadas soluciones a problemas de programación teniendo siempre el mismo formato de uso.

Así, en base a las teorías anteriores, se llevó a cumplimiento cada uno de los objetivos propuestos al inicio de este trabajo, se identificaron los planes de programación básicos

de uso común en la solución de problemas de programación básica, se diseñaron ejemplos resueltos y ejercicios por completar, además se usó como herramienta de apoyo un software.

El estudio se aplicó a un grupo de 42 estudiantes de nuevo ingreso a la carrera de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes. Los resultados obtenidos se evaluaron con distintas técnicas estadísticas mismas que dan evidencia de un resultado favorable en dichos estudiantes.

Se puede señalar que, si es posible aplicar estrategias didácticas donde basándonos en la teoría de la carga cognitiva, y que ésta nos apoya en la elaboración de los materiales didácticos los cuales facilitan la adquisición de habilidades para la resolución de problemas de programación en los estudiantes de nuevo ingreso a esta área del conocimiento.

# RESUMEN INGLES

Taking as reference the cognitive load, this research work aims to design, develop and test teaching materials based on identification of programming plans, this with the purpose of measuring its effectiveness in learning programming in first-year students of the Educational Program in Computational Systems Engineering of the Autonomous University of Aguascalientes.

The basic theories on which this study is based are: Theory of Cognitive Load, Problems to be completed and Programming Plans.

The theory of cognitive load has is based on two ideas. The first indicates that the amount of new information processed by the brain is limited, while the second indicates that there is no limit on the amount of information processed by the brain from which it is already stored. The theory of cognitive load supports explicit teaching models, because these models follow the principles of how the brain learns more effectively (Kirschner, Sweller & Clark 2006).

The theory of problems to be completed, Merriënboer & Krammer (1987) points out that students pay more attention to the solved examples when they are given the same examples to complete. A problem to be completed is a partially solved problem in which the student has to complete the solution to it, it is important that when preparing this type of examples to be completed, pay particular attention to the student completing those parts that are intended to form a "scheme" of solution in your long-term memory.

And finally, a programming plan is defined as a sequence of instructions to solve problems that has a solution in the same way always, that is, the solutions follow a uniform plan. A typical case of these are the accumulators, which can be used in different and very varied solutions to programming problems always having the same format of use.

Thus, based on the previous theories, each of the objectives proposed at the beginning of this work was carried out, the basic programming plans of common use in the solution

of basic programming problems were identified, solved examples and exercises to complete were designed, as well as support software to complete the planned method. The study was applied to a group of 42 students of first grades in the career in Computer Systems Engineering at the Autonomous University of Aguascalientes. The results obtained were evaluated with different statistical techniques that give evidence of a favorable result in those students.

It can be noted that it is possible to apply teaching strategies based on the theory of cognitive load, and that this supports us in the development of teaching materials which facilitate the acquisition of skills for solving problems of programming in students of first grades into this area of knowledge.

# TITULO DE LA TESIS

"Diseño y uso de planes de programación como apoyo para la enseñanza de la programación"

# Introducción

## 1.1 Contexto general del problema

El uso de lenguaje de programación para desarrollar aplicaciones es hoy por hoy una habilidad que los estudiantes del área de la computación deben desarrollar, más sin embargo, esto ha probado ser un verdadero reto para estudiantes y profesores; el desarrollo de estas habilidades idóneas para aprender la programación es un tema difícil ( Wang, Fong, Choy & Wong, 2007). Aprender a programar es una actividad abstracta y suele convertirse en un reto para los estudiantes de primer año de las carreras relacionadas con el área de las ciencias computacionales (Jenkins, 2002; Milne & Rowe, 2002; Mow, 2008; Ogar, Shabalina, Davtyan & Kizim, s/f).

La dificultad que representa el aprender a programar, se ve reflejada en el poco aprovechamiento que manifiestan los estudiantes de éstas asignaturas durante sus primeros cursos, así pues, este grado de complejidad que tiene el aprender a programar, se traduce en un nivel de reprobación y muchas veces de deserción por parte de los estudiantes (Wiedenbeck, LaBelle & Kain, 2004).

Oliver (1993) inició una estructura jerárquica en el conocimiento de programación basado en su experimento; de ahí derivó que existen tres tipos básicos de conocimientos en el área de la programación: La comprensión de la sintaxis o estructura del programa, la capacidad de aplicar las declaraciones (esto significa combinar declaraciones y organizarlas en el algoritmo) y la capacidad para resolver problemas. Esto, ha llevado tanto a Oliver como a otros investigadores a plantear nuevos métodos y nuevas herramientas para la enseñanza de la programación.

Stuart Garner ( 2001) comenta que existen diversos trabajos sobre los métodos acerca de cómo enseñar a programar a estudiantes de los primeros años de carreras del área de las ciencias computacionales. Estos trabajos, muestran ideas y propuestas y cada una de ellas ha alcanzado un diferente grado de éxito; más sin embargo aún los estudiantes encuentran este proceso complicado.

Stuart Garner (2001) señala que el proceso de aprender a programar se torna complejo debido a que el estudiante debe desarrollar diferentes habilidades de manera simultánea, como son la resolución de problemas, el diseño de algoritmos, diseño de los programas y el lenguaje de programación que se esté usando. Aunado a esto, los estudiantes deben aprender técnicas que les permitan probar y depurar sus códigos, así como modificar los mismos en caso de así requerirlo.

Existe un conjunto de elementos que se deben tomar en cuenta para diseñar los planes de programación, esos son el lenguaje de programación que se va a usar para enseñar el proceso de enseñanza-aprendizaje, el ambiente de desarrollo adecuado, la metodología de enseñanza-aprendizaje a implementar, los libros de texto que más apoyen al estudiante a avanzar en el dominio de la materia, entre otros (Xinogalos, 2014).

De la misma manera, es importante tomar en cuenta que el mundo de la tecnología avanza cada vez más rápido y es necesario adaptar nuevos mecanismos que ayuden a los estudiantes a adaptarse más y mejor a estos cambios, así pues, es importante utilizar métodos de enseñanza innovadores que permitan a los cursos de programación adaptarse a estas necesidades.

En lo que se refiere a los estudiantes de la carrera de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes, se ha observado un fenómeno similar al ya descrito, los estudiantes de los primeros cursos relacionados con la programación han sufrido a lo largo de los años un índice considerable de reprobación y de desercion, muchas de las veces por considerar que estas materias tienen un grado de complejidad alto.

En relación a lo anterior, es que se pretende desarrollar y probar un método alternativo que apoye a los estudiantes de dichas carreras a transitar por este proceso de aprendizaje de una mejor forma, este método consta de planes de programación, ejercicios resueltos, ejercicios por completar y un software que apoye en el desarrollo de los algoritmos y la escritura de los algoritmos en un lenguaje de alto nivel, esta metodología se desarrollará con el fin de ser un apoyo didáctico para el tutor de materias relacionadas con el área de la programación.

## 1.2 Estudios previos

Wiedenbeck (2004) concluyó que "*las decisiones sobre la especialización en ciencias de la computación y campos relacionados, a menudo se determinan por el éxito o el fracaso de un estudiante en el curso introductorio. Si un estudiante abandona, falla o pasa con una lucha, es poco probable que el estudiante se inscriba para un curso de continuación. A pesar de la investigación sobre los factores que influyen en la inscripción y el éxito de los novatos en la programación introductoria, todavía no se entiende completamente lo que hace un curso de programación introductoria positivo y exitoso para algunos, pero difícil y frustrante para los demás*".

Se han realizado diversos estudios con el fin de determinar cuáles son las dificultades y retos a los que se enfrentan los estudiantes de programación principiantes. A continuación, se enuncian algunos de ellos.

Tavares *et al*. (2001) señalan que la organización del currículo y los métodos de enseñanza son los dos factores que más afectan los índices de fracaso en los cursos de programación inicial.

Meisalo *et al.* (2002) encontraron que aproximadamente el 30% de los estudiantes de su curso de programación desertaron por considerar que los ejercicios de programación eran demasiado complejos.

Ala-Mutka (2004:3) señala que "*el aprender a programar conlleva varias actividades, entre ellas aprender las características del lenguaje, el diseño del programa y la comprensión del mismo*". Las técnicas de enseñanza de la programación más comunes que se refleja en los libros de texto, es comenzar por la sintaxis del lenguaje de programación que se pretende aprender, pasando posteriormente a la semántica del mismo y prestan poca atención a las estrategias de aprendizaje que se debería utilizar para resolver problemas de programación.

Winslow (1996) declara que *"los programadores principiantes conocen la sintaxis y la semántica de las sentencias individualmente, pero no saben cómo combinar estas características en programas funcionales*"

Los estudios anteriores sugieren que aprender a programar requiere que los estudiantes presten la misma atención a los conocimientos de programación (sintaxis y semántica), así como a las estrategias de resolución de problemas (Iqbal & Harsh, 2013).

### 1.3 Problemática particular

Dentro del contexto de las carreras que tienen relación con la ciencias computacionales, el aprender a programar con el fin de desarrollar aplicaciones, es una habilidad que los estudiantes deben desarrollar, más sin embargo, varios investigadores como Wang, Fong, Choy, Chang, Hsiao, Van Merriënboer y De Croock, (Chang, Chiao & Hsiao, 1996; Van Merriënboer, 1990; Wang *et al*., 2007) entre otros, señalan que "*el adquirir estas habilidades representa un verdadero reto para los estudiantes de primeros semestres. Aprender a programar conlleva mucho más que sólo aprender una serie de instrucciones y el acomodo de las mismas, el aprender a programar, es una actividad en sí misma abstracta y creativa, lo que la hace una actividad compleja*".

Investigadores tales como Jenkins, Milne & Rowe, Mow, Ogar (Jenkins, 2002; Milne & Rowe, 2002; Mow, 2008; Ogar et al., s/f) y más, señalan que "*el aprendizaje de la programación ha probado ser compleja*", esto se ve reflejado en el poco aprovechamiento que reflejan los estudiantes en las materias relacionadas con ello, esto

tiene un impacto directo en el número de alumnos que reprueban estas materias o incluso desertan de las carreras que tiene que ver con ello (Wiedenbeck et al., 2004).

De la misma manera Shuhidan (2012) señala que "*aprender a programar se considera una tarea difícil y desafiante para un gran número de principiantes en esta área*", por ello las tasas de reprobación y deserción se tornan altas ente un 30 a 50% según sugiere un estudio (Guzdial & Soloway 2002, Kinnunen & Malmi 2006).

Así pues, se puede observar que esta problemática se ha estudiado desde hace mucho tiempo, ya que no sólo es un problema en México, y particularmente en Aguascalientes, sino que es un problema detectado de manera global en todos los programas educativos de área de las ciencias computacionales; llevando así a múltiples investigadores a buscar y probar distintas formas y medios que apoyen en el proceso de enseñanza aprendizaje en este tema, los estudios han incluido temas como la parte cultural (Bruce, 2004), diversos instrumentos de apoyo (Kelleher & Pausch, 2005), incluso temas como el razonamiento o la conciencia (Bornat, Dehnadi & Simon, 2008) y la metacognición (Parham, Gugerty & Stevenson, 2010).

En lo que respecta a los programas académicos relacionados con esta área en la Universidad Autónoma de Aguascalientes, a lo largo de los últimos años, se ha podido observar en múltiples grupos de programación cómo desarrollar las habilidades necesarias para dominar un lenguaje de programación significa un verdadero reto, tanto por parte de los estudiantes como por parte del docente. El proceso de desarrollo de habilidades para el dominio de los leguajes de programación, incluye diversas destrezas que se debe de desarrollar en los estudiantes, así como la selección de una estrategia para una primera aproximación a la enseñanza de la programación que les sea de ayuda y les provea de habilidades para resolver problemas.

# PROBLEMA DE INVESTIGACIÓN:

## 2.1 Objetivo general

Diseñar, desarrollar y probar material didáctico basado en identificación de planes de programación con el propósito de medir su efectividad en el aprendizaje de la programación en alumnos de primer año del programa educativo de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes.

## 2.2 Objetivos específicos:

- Identificar planes de programación básicos de uso común en la solución de problemas de programación.
- Desarrollar material didáctico basado en los planes identificados.
- Probar los planes desarrollados en un grupo como prueba piloto.

## 2.3 Preguntas de investigación

¿Es posible identificar y documentar un conjunto de planes de programación, que sirvan como material didáctico para estudiantes que inician en esta área de conocimiento?

¿El hacer explícitos planes de programación al diseñar material didáctico de programación básica, proporciona a los estudiantes un mayor número de habilidades para la solución de problemas de programación?

¿Es posible aplicar estrategias didácticas meta cognitivas a planes de programación, para facilitar la adquisición de habilidades de resolución de problemas de programación?

## 2.4 Justificación:

En todo ambiente de enseñanza-aprendizaje, las técnicas, herramientas, actividades, son la base para llevar a buen término dicho procesos. Es indispensable que los estudiantes adquieran nuevas habilidades o mejoren aquellas con las que ya cuentan, sin embargo cada estudiante tiene distintas formas de aprender (Gallego-Durán, Molina-Carmona & Llorens-Largo, 2016).

Las metodologías de enseñanza son los métodos o enfoques que los tutores adoptan al momento de preparar material de enseñanza para los estudiantes (Mohorovicic & Strcic 2011). Papp Varga y col (2008) argumentan que la enseñanza de las TIC's es un nuevo tipo de problemática en comparación con las matemáticas o la física, debido a esto su metodología de enseñanza aún no está bien establecida o desarrollada y como consecuencia, cada tutor adopta sus propios métodos de enseñanza.

Pashler *et al.* (2008) sugiere que "*estilos de aprendizaje se refiere al concepto que los individuos difieren en cuanto a qué modo de instrucción o estudio es más eficaz para ellos*". Fleming y Baume (2006) introdujeron el modelo VARK de aprendizaje mismo que define los estilos de aprendizaje en visual, auditivo, de lectura / escritura y de aprendizaje quinestésico. Este modelo reconoce que los estudiantes tienen diferentes enfoques para procesar información a lo cual se le conoce como su modo de aprendizaje preferido. Los estudiantes con un estilo de aprendizaje visual absorben la información al verla, los aprendices auditivos aprenden mejor escuchando y hablando, los aprendices aprenden mejor haciendo y los aprendices de lectura / escritura prefieren aprender leyendo libros y folletos.

Así pues, existen estudiantes, a quienes se les facilita más el desarrollo de ejercicios o prácticas por su cuenta, otros a los que les es más sencillo tomar una clase tradicional donde el profesor es el actor que se encarga de trasmitir conocimientos. Para otros estudiantes resulta más fácil estudiar por su cuenta mediante libros, y ahora muy de auge a través de videos, pero también existen quienes aprenden más mediante el uso repetido de determinadas actividades que les ayude de alguna manera a grabar en su mente un proceso repetitivo; este último es el campo en dónde se pretende trabajar mediante el "Diseño y uso de planes de programación" que les permitan a los estudiantes desarrollar las habilidades necesarias para dominar la programación.

Según Chang, Chiao & Hsio (Chang et al., 1996a), una plantilla, "*es una unidad de construcción básica para implementar un programa*", ellos mencionan que al aprender los estudiantes una plantilla específica, lo que realmente están aprendiendo es el concepto fundamental que ésta representa. Así mismo menciona, que con el uso de las

plantillas los estudiantes se familiarizan con la solución de pequeños problemas, y esto les ayuda a que posteriormente resuelvan problemas más grandes.

En su estudio Chang y Hsiao (1996a), probaron mediante un experimento que el sistema de completado de plantillas tiene un efecto benéfico en las técnicas de enseñanza de la programación con estudiantes de primeros semestres, Chang y Hsiao, demostraron mediante un experimento que los estudiantes que usaban las técnicas de completado de plantilla, tenían menos conflictos para encontrar solución a problemas convencionales, también probaron que el uso de las plantillas de técnicas de completado de solución de problemas, logra en los estudiantes más y mejores habilidades para la solución de problemas comunes mediante la programación; obteniendo así habilidades que posteriormente los lleve a dominar cualquier lenguaje de programación.

Van Merrienboer y De Croock (1992) en uno de sus estudios, concluyeron que el uso de completado de plantillas de programación se pude utilizar eficazmente para enseñar a programar a los estudiantes de primeros semestres, y sugirieron que el uso de la estrategia de completado de platillas podría llegar a ser importante para el diseño del sistema de tutoría para el aprendizaje de cursos de programación para principiantes.

# MARCO TEÓRICO

## 3.1 Introducción

Se han propuesto distintos métodos para mejorar el aprendizaje de los estudiantes tales como aprendizaje basado en problemas, casos de estudio y el aprendizaje como tal, estos tres elementos se pueden integrar para llevar a cabo el proceso de enseñanza/aprendizaje de la programación; al aplicar estos métodos conjuntos se pretende obtener un método de enseñanza más fácil y eficaz, el cual los autores Kölling & Barnes (2008) señalan da mejores resultados.

## 3.2 Teoría de carga cognitiva

La teoría de la carga cognitiva (Centre for Education Statistics and Evaluation, 2017) tiene dos ideas base. La primera indica que la cantidad de información nueva que procesa el cerebro es limitada, mientras que la segunda se indica que no hay límite en la cantidad de información procesada por el cerebro de la cual ya se tiene almacenada. Esta teoría, tiene su base un gran número de teorías muy aceptadas que se basan en como el cerebro humano almacena y procesa la información (Cierniak , Scheiter & Gerjets 2009, p. 44). Ellos indican que la memoria se pude dividir en dos, a saber, memoria de trabajo (o memoria de corto plazo) y memoria de largo plazo.

La información es almacenada en la memora de largo plazo en lo que se conoce como esquemas los cuales organiza elementos de información de acuerdo a cómo serán usados. La teoría de esquemas, indica que, para obtener una habilidad cognitiva, es necesario construir esquemas cada vez más complejos, esto se logra agregando cada vez más esquemas básicos a los esquemas más altos y no hay un límite que indique que tan complejo puede ser un esquema. Una parte muy importante en la construcción del esquema es la automatización, esto es que tan fácilmente y automáticamente puede ser procesada la información. El proceso de automatización en el procesamiento de la información y se da **con la practica exhaustiva de una tarea** (Sweller, van Merrienboer & Paas 1998, p. 256).

Sweller y Cooper (1985) señalan en su artículo que al usar ejemplos con los alumnos para la enseñanza del algebra y hacerles que ellos los estudiaran y los comprendieran para posteriormente explicarlos, disminuye en ellos la carga de trabajo cognitivo al intentar comprender partes de la solución de un problema y posterior a esta fase resolver ejercicios en los cuales la solución involucra todas las partes aprendidas previamente.

De manera similar a como se obtiene la habilidad para tener más y mejores movimientos en el ajedrez mediante la practica continua según señalan John Sweller, Paul Ayres y Salva Kalyuga en su libro Cognitive Load Theory (2011), así de una forma similar sucede con los estudiantes, con la práctica continua estos adquieren más y mejores herramientas de solución a problemas de cualquier tipo.

Señalan además que esta práctica continua genera cambios cognitivos debido a que lo que se aprende durante la práctica *se almacena en la memoria a largo plazo*, así pues, el realizar practica continua de solución de problemas y ejercicios genera en los estudiantes un aumento en la habilidad de solución a problemas planteados y genera en ellos un conocimiento más perdurable.

De esta manera, una vez que se ha obtenido el conocimiento de una solución que se ha practicado más una vez, y que se ha convertido en conocimiento para ellos, cuando se les presenta un problema que requiere de una solución similar, es posible acceder al conocimiento generado previamente para obtener la solución requerida.

Se ha encontrado que la memoria a largo plazo contiene muchos **esquemas** y que estos de alguna manera determinan como es que se procesa la información que nos llega. Lo que vemos y oímos no está determinado únicamente por la información que afecta a nuestros sentidos, sino en gran medida por los esquemas almacenados en la memoria a largo plazo.

Así pues, se puede decir que un esquema nos ayuda a clasificar elementos y a obtener soluciones más rápidas cuando se ha pasado por un proceso de aprendizaje repetitivo.

Un esquema se puede definir pues como una construcción cognitiva que nos permite clasificar múltiples elementos de información en un solo elemento y podríamos decir que un estudiante puede resolver problemas debido a que probablemente haya generado un esquema para ello.

De lo anterior se pude pues inferir que cuando un estudiante ha generado en su memoria de largo plazo un esquema de resolución a cierto tipo problemas, comienza a tratar de una manera similar las soluciones para problemas futuros similares ya que esta nueva solución de alguna manera es similar a la solución del esquema generado.

Debido a los esquemas generados en la memoria a largo plazo de las personas, nos es más fácil y posible resolver problemas que muy probablemente de no haberse generado ese esquema en nuestra memoria, no podríamos resolver.

Los esquemas recién adquiridos deben procesarse de forma consciente y algunas veces con un esfuerzo considerable, pero con la constante practica se adquieren nuevos esquemas en nuestra vida cotidiana y de manera similar en la vida académica, así para ser más competitivos en las áreas de conocimiento, deberíamos adquirir más esquemas.

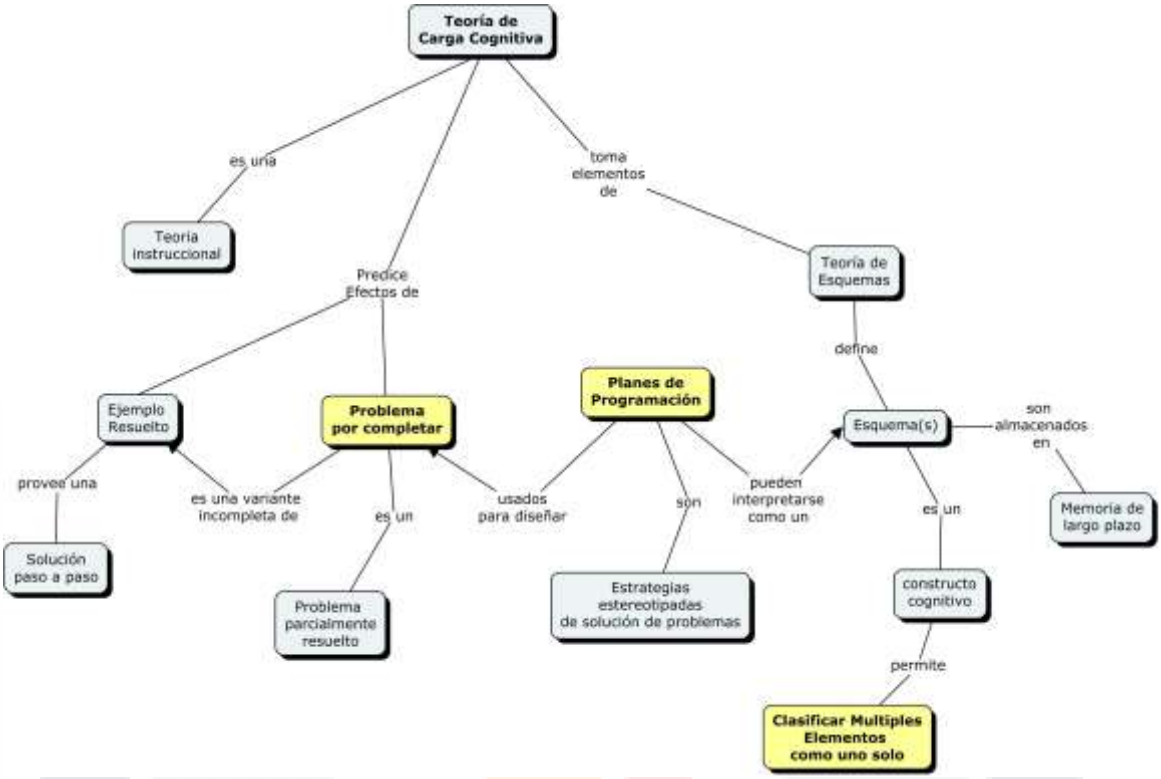



*Ilustración 1: Teoría de Carga Cognitiva*

La teoría de la carga cognitiva tiene su respaldo en un número suficiente de estudios de control aleatorios que han dado muestra de que la enseñanza tiende a ser más efectiva cuando ésta basa su diseño en como el cerebro humano procesa y almacena la información.

La teoría de la carga cognitiva como se pude apreciar en la ilustración 1, es una teoría instruccional que toma elementos de la teoría de esquemas, los cuales se forman en la memoria de largo plazo. Predice efectos en estudios que utilizan ejemplos por completar y la teoría del ejemplo resuelto mismas que proveen una solución paso a paso a un problema dado.

### 3.3 Problemas por completar

Un enfoque de enseñanza que forma parte de los efectos predichos por la teoría de carga cognitiva y que tiene respaldo en un número considerable de estudios es el que lleva por nombre "*efecto del ejemplo resuelto*". Por ejemplo, en un estudio realizado por Cooper y Sweller (1987) diseñaron un experimento en el cual los alumnos de una determinada institución debían aprender a resolver distintos problemas simples de algebra, en este experimento, encontraron que los alumnos que se les había enseñado con el uso de distintos ejemplos resueltos aprendieron más rápido que los alumnos que tenían que dedicar tiempo a resolver por ellos mismos los problemas. Así mismo, encontraron que aquellos alumnos que habían utilizado el método de aprendizaje mediante ejemplos resueltos, no solo eran más capaces en la solución de problemas de la misma índole en tiempo posterior, sino que también fueron más capaces de transferir este conocimiento a nuevos problemas por resolver donde las reglas matemática eran las mismas pero aplicada a contextos o problemas distintos. Este mismo efecto se ha replicado en varios estudios ente los cuales se puede citar: Pass (1992), Paas & van Merrienboer (1994), Pillar (1994), Quilici & Mayer (1996), Tuovinen & Sweller (1999), entre otros.

Crissman (2006) encontró que este tipo de herramientas usadas en los estudiantes daba un efecto del 52%, lo que nos da una muestra del impacto de esta técnica aplicada en el proceso de enseñanza aprendizaje en los estudiantes.

El "*efecto del ejemplo resuelto*" está relacionado con varios efectos instruccionales importantes y en partícula con el efecto de ejercicios por completar.

Merriënboer & Krammer (1987) señalan que los alumnos prestan una mayor atención a los ejemplos resueltos cuando se les proporciona a los mismos ejemplos por completar. Un problema por completar en un problema parcialmente resuelto en el cual el estudiante tiene que completar la solución al mismo, es importante que cuando se elaboran este tipo de ejemplos por completar, se ponga particular atención en que el estudiante

complete aquellas partes que se pretende formen un "*esquema*" de solución en su memoria a largo plazo.

Van Merriënboer (1990) realizó un primer estudio utilizando problema por completar tenido como base la teoría de la carga cognitiva, este estudio fue aplicado a un grupo inicial en el aprendizaje de la programación. Durante un conjunto de lecciones los estudiantes se apegaban a una metodología tradicional donde se les pedía que diseñaran y codificaran la solución a un problema planteado o bien podían utilizar una estrategia de completado de problemas la cual requería la modificación y extensión de programas ya solucionados. Encontró que el grupo que usaba problemas por completar y extensión de los mismos tuvo un desempeño superior que los alumnos que se apegaban al esquema más común de diseñar y codificar el resultado a un problema dado (John Sweller, Paul Ayres y Salva Kalyuga, 2011 p: 1052-106).

Así mismo, Van Merriënboer y de Croock (1992) realizaron un estudio similar con contenido de programación. Ellos compararon un grupo que llevaba una instrucción tradicional y un grupo que usaba problemas por completar. Los resultados de su estudio indicaron claramente un aprovechamiento superior en el grupo que utilizo problemas por completar donde se les presentaba a los estudiantes ejercicios incompletos y ellos deberían de completar la solución al problema planteado.

Stuart Garner ( 2001) señala que diversos investigadores han usado métodos de problemas por completar y diferentes ejemplos prácticos para la enseñanza de la programación en lugar de las instrucciones que se usan comúnmente, y han encontrado en esta forma de enseñanza grandes beneficios y ventajas en los estudiantes que han usado estos métodos.

Así mismo, Van Merrienboer, Krammer y Maaswinkel (1994a) señalan que cuando los estudiantes se enfocan en leer soluciones parciales a problemas, para posteriormente ellos poder completar la solución planteada, esto hace que la lectura, la comprensión, el modificar y ampliar estos problemas les dé más y mejores herramientas para aprender los puntos clave que van completando en los problemas propuestos, pero para lograr

este aprendizaje significativo es necesario que los problemas estén bien diseñados y sigan un orden de aprendizaje que conduzca a los estudiantes a lograr cada vez más y mejores esquemas, de la mano con más y mejores resultados en la resolución de problemas.

Existen además otros factores que pudiéramos tomar en cuenta en la enseñanza de la programación para estudios posteriores, como puede ser el alto componente de creatividad que ésta actividad implica y la cual a su vez existe estudio en otros contextos como el estudio de la ciencia, la tecnología, la ingeniería, el arte y las matemáticas (Park & Ko, 2012).

### 3.4 Planes de programación

El método de leer ejemplos o ejercicios de programación ya resueltos se ha tomado como un método de enseñanza por varios investigadores y se ha puesto en práctica con resultados favorables. Van Merrienboer y Krammer (1987; Merriënboer Krammer y Maaswinkel, 1994) señalan que el leer ejercicios resueltos les permite tener una mejor comprensión a la hora de solucionar problemas de programación. Sin embargo, también señala que el sólo leer códigos ya solucionados, no es suficiente para la enseñanza de la programación, debido a que los estudiantes no pueden abstraer de la lectura todas las posibles soluciones o los **planes** de solución que estos manejan, **un plan de programación se define como una secuencia de instrucciones para solucionar problemas que se desarrollan de la misma manera siempre, es decir, las soluciones siguen un plan uniforme**, un caso típico de éstos son los acumuladores, mismos que se pueden usar en distintas y muy variadas soluciones a problemas de programación teniendo siempre el mismo formato de uso.

Se ha desarrollado un sistema automatizado para la planificación y construcción de tareas para la programación introductoria. Estas tareas consisten en ejemplos de programas incompletos para que el estudiante los complete, ejemplos para que el estudiante amplié o modifique códigos ya hechos y características sobre el funcionamiento y la estructuración básica de un programa. El modelo se basa en planes de programación, el perfil del estudiante a quien va dirigido y la base del problema a

resolver, esta idea de los planes de programación como apoyo a la solución de problemas se sustenta en la teoría de planes de programación usada por Van Merrienboer, Krammer y Maaswinkel (1994).

Van Merriënboer (1990)(Van Merriënboer & De Croock, 1992) en un estudio menciona que hay 3 operaciones básicas en el proceso de aprendizaje **por completado de planes**, estas son: rellenar los espacios en blanco, modificar el código proporcionado y extender el código proporcionado, señala que los espacios en blanco representan los conceptos claves que el tutor quiere que los estudiantes aprendan, de aquí la importancia de diseñar los planes persiguiendo un objetivo claro.

# 4 DISEÑO DE LA SOLUCIÓN

Con base a las teorías antes mencionadas y de cómo estas han demostrado tener éxito en diversos experimentos, y en base a la experiencia docente en el área de la enseñanza de la programación, se identificaron los planes de programación básicos para los estudiantes de nuevo ingreso al área de la programación.

Una vez identificados estos planes, se diseñaron cada uno de ellos en conjunto con un grupo de ejercicios resueltos y ejercicios por completar, se identificó un software que proporcione un apoyo para el aprendizaje de la programación en estudiantes iniciales en este tema, para ello se seleccionó el software PSeInt.

A continuación, se detalla cada uno de los planes identificados acompañados de ejercicios resueltos, ejercicios por completar y apoyo del software seleccionado.

## 4.1 Estructuras selectivas

**Descripción:**

Las estructuras de control selectivas, se usas en la solución de casi todo tipo problemas a resolver. Se utilizan cuando es necesario tomar una decisión de hacia dónde continuar, o marcar un camino a seguir.

Estas decisiones se toman en base a la evaluación de una o más condiciones que nos indicarán el camino a seguir.

Existen numerosas ocasiones en las cuales las decisiones se pueden tomar siguiendo un conjunto de dependencias, es decir que una vez que se toma un camino, se evalúa un camino siguiente y se toma la decisión de hacia dónde seguir, este proceso se puede seguir por un número indeterminado de veces, en este caso lo que se aplica o se sigue como solución a un problema es un tipo árbol de decisiones que nos lleva a una solución específica.

Las estructuras de selección más usadas para la toma de decisiones en los lenguajes de programación, se pueden categorizar como sigue:

- SI – ENTONCES (condición simple)
- SI – ENTONCES / SINO (condición selectiva doble)
- SI MULTIPLE (Estructura selectiva usada para múltiples casos de selección)

Cuando es necesario, múltiples decisiones se pueden combinar para generar una solución más compleja y así, éstas forman un tipo árbol de decisión, las estructuras condicionales se combinan según la necesidad de a donde se quiere llegar como camino de solución final.

A continuación, se describe cada una de ellas y se propone un plan de programación para la instrucción de la misma.

### 4.1.1 Plan condicional simple

**Descripción:** Este plan permite que el flujo del programa siga por un camino específico, si se cumple con una condición o conjunto de condiciones determinadas, esta condición o condiciones se plasman mediante una pregunta, misma que es la que determina el camino que se debe seguir a través de las respuestas afirmativas. Si el resultado de la pregunta es verdadero, entonces se entra a la o las instrucciones que formen parte de dicha condición, de lo contrario se salta estas instrucciones y se continúa con el flujo normal del programa (ilustración 2).

**Estructura del plan condicional simple:**

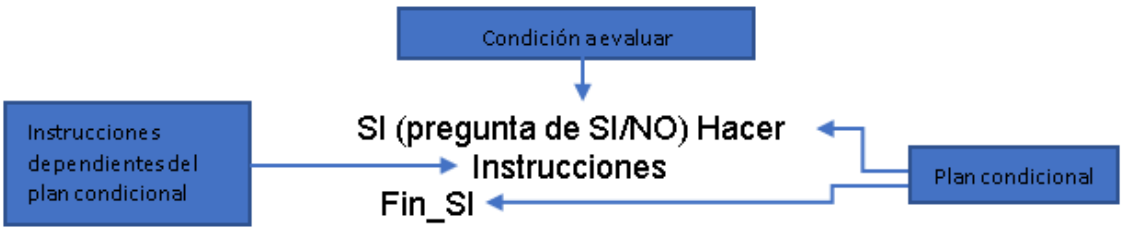



*Ilustración 2: Plan condicional simple*

**Ejemplo 1 de Plan de programación "Condicional Simple" (Ilustración 3):** Se requiere analizar la calificación de un alumno en un examen, es necesario imprima la palabra "aprobado" en caso de que esa calificación sea mayor a 8, el ejercicio deberá pedir la calificación en una variable llamada CAL, analizarla y realizar lo solicitado.
La ilustración que se muestra a continuación, da solución al problema planteado.


*Ilustración 3: Ejemplo 1 condicional simple*

En la ilustración 4 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación simple.

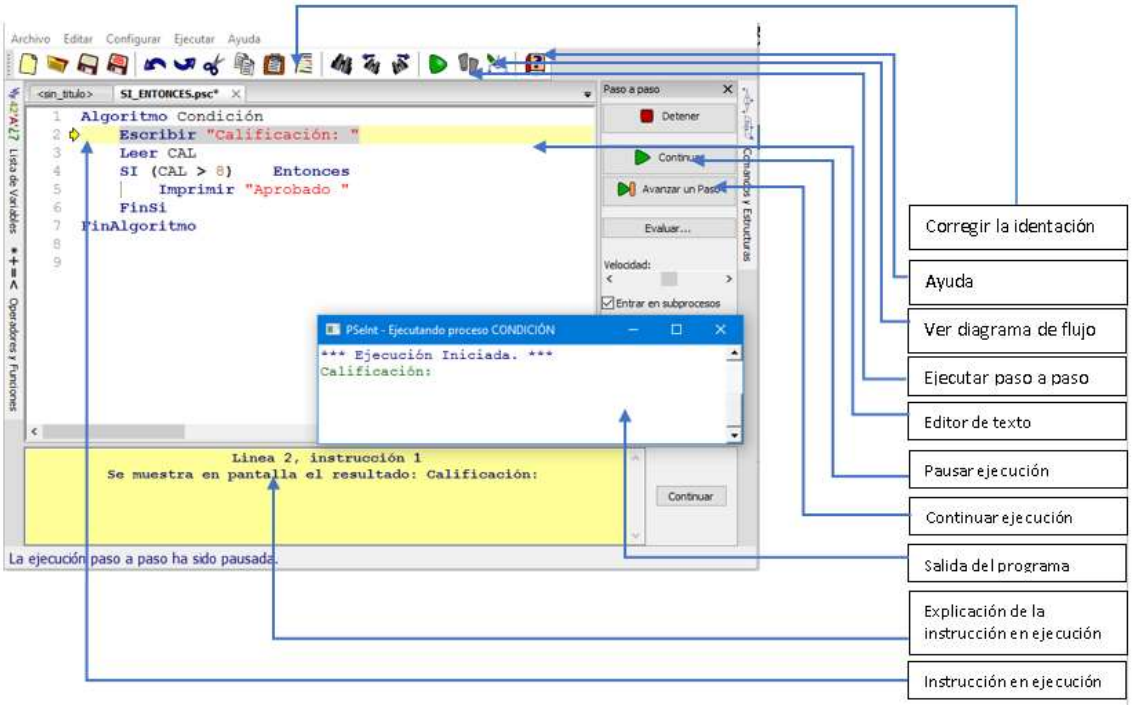



*Ilustración 4: Solución PSeInt ejemplo 1 condición simple*

**Ejemplo 2 de Plan de programación "Condicional Simple":** Se tiene el sueldo de un trabajador en una variable llamada SUELDO, a este sueldo se le debe sumar una 15% de aumento en caso que el sueldo sea menor a 1000, una vez aplicado el aumento imprimir el nuevo sueldo, este ejemplo se resuelve como lo muestra la ilustración 5.

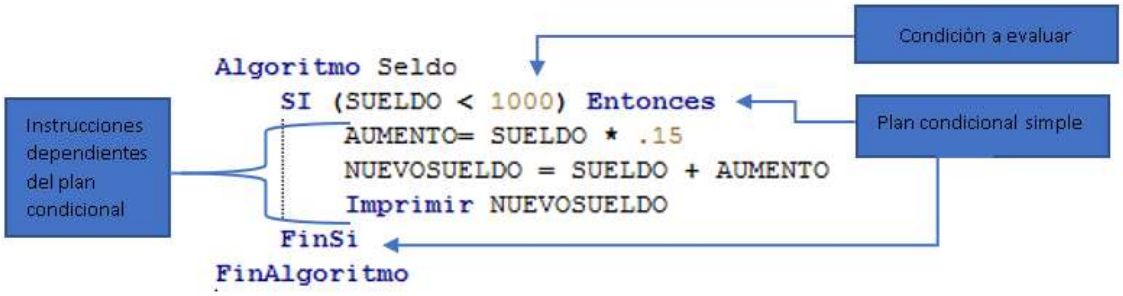



*Ilustración 5: Ejemplo 2 condicional simple*

En la ilustración 6 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación simple.

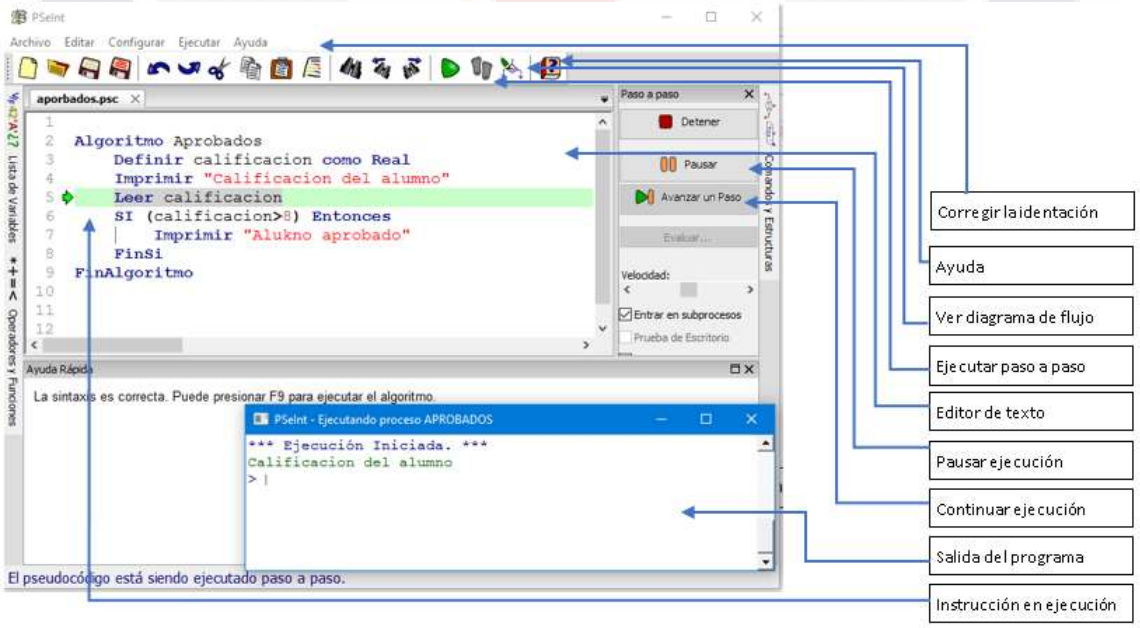



*Ilustración 6: Solución PSeInt ejemplo 2 condición simple*

**Ejercicio 1 de Plan de programación "Condicional Simple":** El siguiente algoritmo, intenta obtener la temperatura en grados Fahrenheit a partir del número de sonidos emitidos por un grillo, ya que se dice que el número de sonidos emitidos por un grillo en un minuto, están definidos en función de la temperatura y es posible determinar el nivel de la temperatura haciendo uso de un grillito como termómetro. La fórmula para lograrlo está dada por: T = N / 4 + 40

Donde:
T es la temperatura en grados Fahrenheit
N es el número de sonidos por minuto del grillo.

**Recuerde que para que un valor pueda ser dividido debe ser mayor a 0.**

La ilustración 7 muestra una solución parcial al problema planteado, escriba en la línea la instrucción que falta para logar que se realice de manera **correcta** la solución al problema antes planteado.

```
Algoritmo SI_ENTONCES
    Escribir "Sonidos por minuto emitidos por el grillo"
    Leer N
    SI _________________ Entonces
        T= N / 4 + 40
        Imprimir "la temperatura es; " T
    FinSi
FinAlgoritmo
```

*Ilustración 7: Ejercicio 1 por completar condicional simple*

### 4.1.2 Plan condicional doble:

**Descripción:**

La estructura SI – ENTONCES /SINO, tiene dos ramas, la primera, verifica si la pregunta que se hace es VERDADERA (ENTONCES) y la segunda verifica si la pregunta que se hace es FALSA (SINO), se verifica la prueba evaluativa, y de salir verdadera se procede

a seguir con las instrucciones de la parte de ENTONCES, mientras que si la prueba a evaluar es negativa se procede a ejecutar las instrucciones de la parte SINO. Sea cual sea el resultado, al terminar las instrucciones pertenecientes a la parte VERDADERA O FALSA, se prosigue con el flujo normal del programa, en la ilustración siguiente se muestra la estructura general para la condición doble.

**Estructura del plan condicional doble:**


*Ilustración 8: Plan condicional doble*

**Ejemplo 1 de Plan de programación "Condicional Doble":** Leer dos números y determinar cuál de ellos es el mayor

La ilustración que se muestra a continuación, da solución al problema planteado.

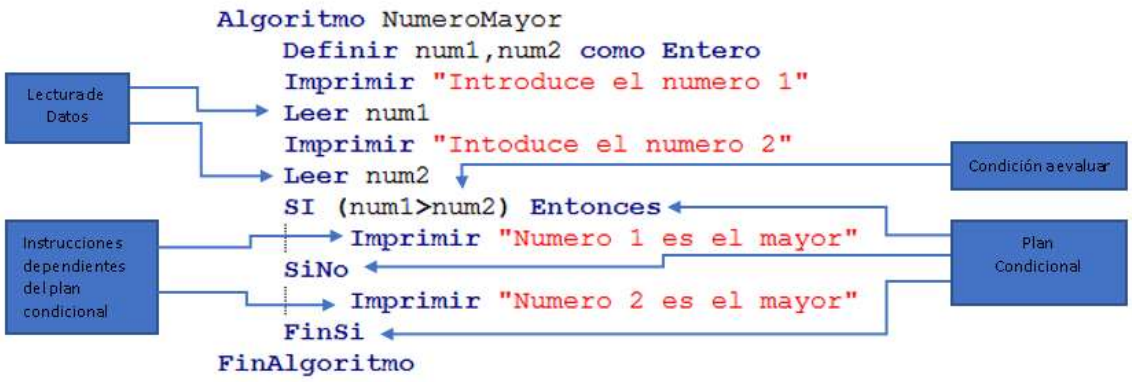



*Ilustración 9: Ejemplo 1 plan condicional doble*

En la ilustración 10 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación doble.

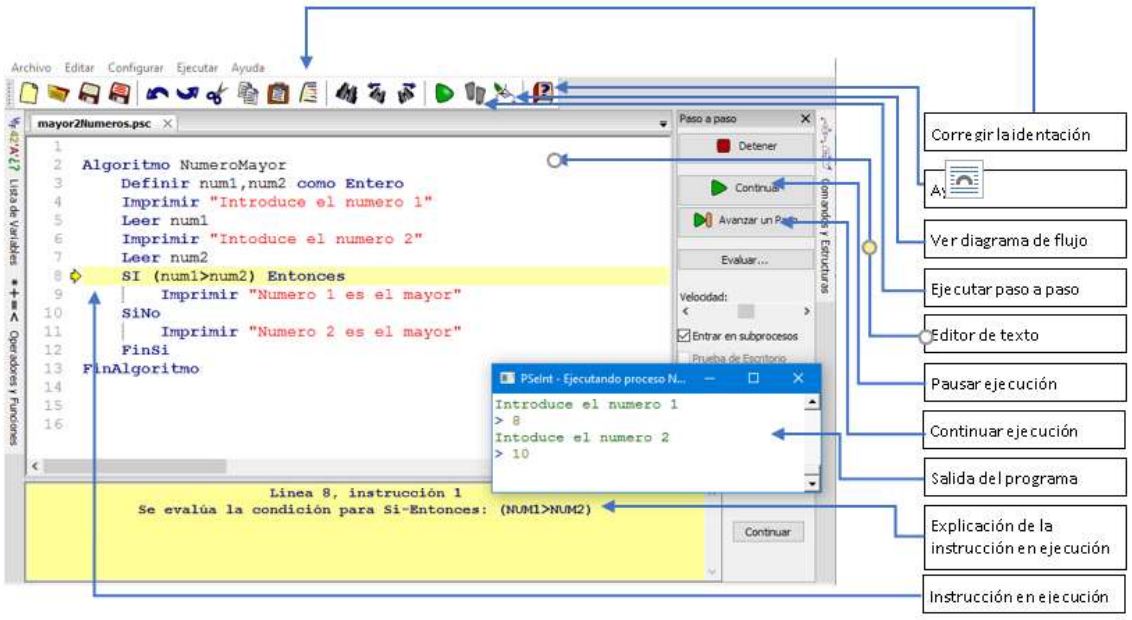



*Ilustración 10: Solución PSeInt ejemplo 1 condicional doble*

**Ejemplo 2 de Plan de programación "Condicional Doble":** Analizar la calificación obtenida por un alumno en un examen, imprimir la palabra "aprobado" en caso de que esa calificación sea mayor a 8 y en caso contrario imprimir la palabra "reprobado", la calificación es leída en una variable llamada CAL.

La ilustración que se muestra a continuación, da solución al problema planteado


*Ilustración 11: Ejemplo 2 condicional doble*

En la ilustración 12 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación doble.

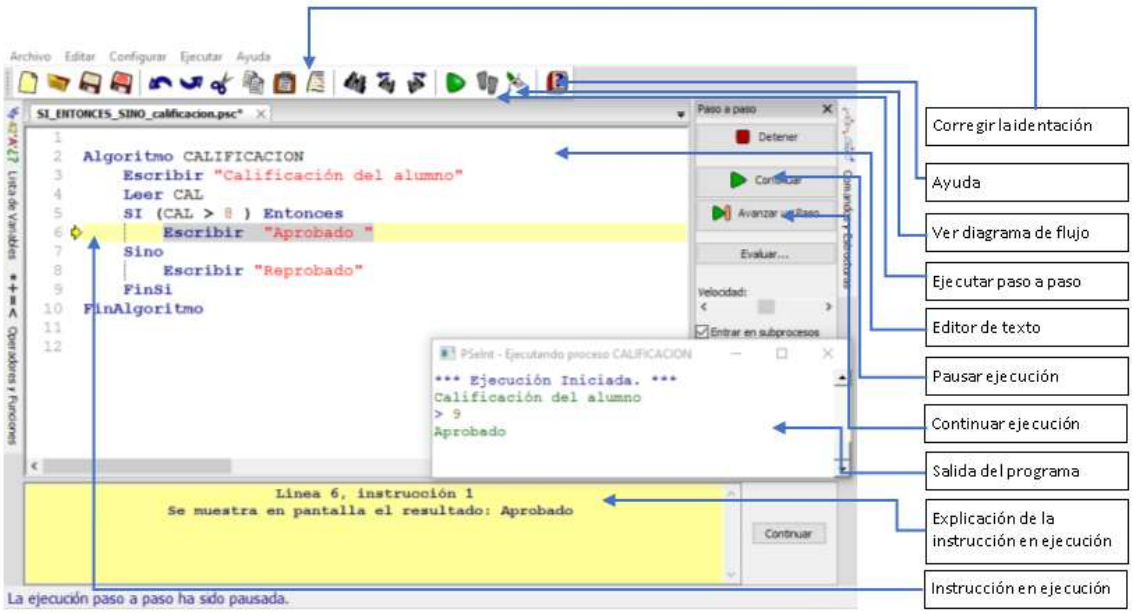



*Ilustración 12: Solución PSeInt ejemplo 2 condición doble*

**Ejemplo 3 de Plan de programación "Condicional Doble":** Se tiene el sueldo de un trabajador en una variable llamada SUELDO, a este sueldo se le debe sumar un 15% de aumento en caso que el sueldo sea menor a 1000 y un 12 % en caso contrario, una vez aplicado el aumento imprimir el nuevo sueldo.

La ilustración siguiente da solución al problema planteado.

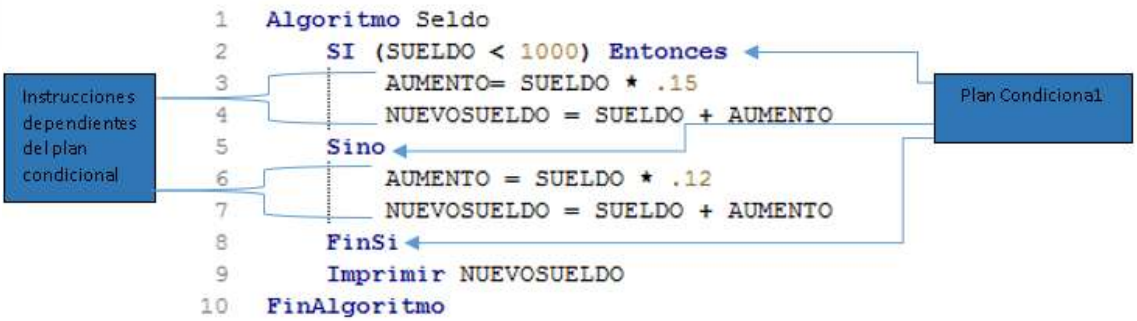



*Ilustración 13: Ejemplo 3 condicional doble*

En la ilustración 14 se muestra la solución en el software **PSeInt** al ejemplo 3 del plan de programación doble.

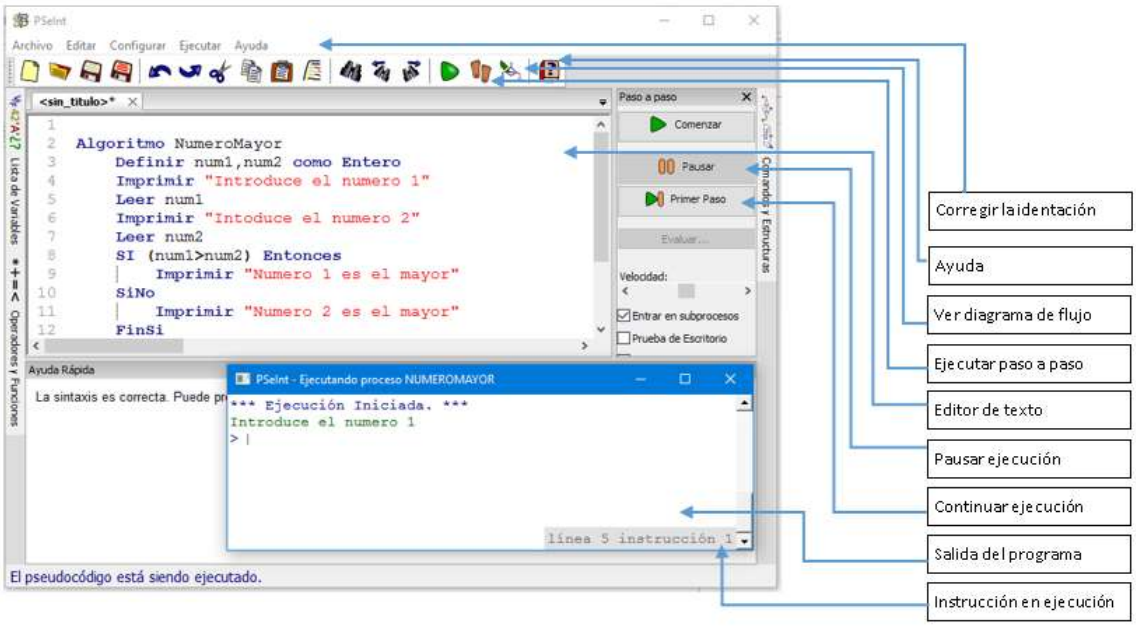



*Ilustración 14: Solución PSeInt ejemplo 3 condición doble*

**Ejercicio 1 de Plan de programación "Condicional Doble":** Se requiere de un algoritmo que, dado un número entero, determine e imprima si el mismo es positivo, negativo o nulo.

La ilustración 15 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
1   Algoritmo SI_ENTONCES_SINO
2       Leer NUMERO
3       SI_____________Entonces
4       |   Imprimir "Numero positivo "
5       _________
6       |   SI_____________Entonces
7       |   |   Imprimir "Numero negativo"
8       |   Sino
9       |   |   Imprimir "Nulo"
10      |   FinSi
11      FinSi
12  FinAlgoritmo
```

*Ilustración 15: Ejercicio 1 por completar condicional doble*

### 4.1.3 Plan condicional MULTIPLE

**Descripción:** El plan condicional MULTIPLE permite construir un árbol de múltiples decisiones, es decir, se tiene un conjunto de caminos a tomar todos en base a una misma pregunta o una misma variable de decisión y en función del valor que tome la variable de decisión, será el camino por el cual se continúe en la ejecución del programa. En un supuesto que la variable de decisión pueda tomar valores de 1 al 3, si toma el valor de 1, se irá por un camino, si toma el valor de 2 por otro camino y si adopta el valor de 3 por un camino más, cada uno de ellos llevarán a la salida del plan condiciona de casos múltiples, además, esta estructura permite un caso por defecto, es decir si la variable no adoptara ninguno de los valores que se han definido, se va por la opción por defecto, misma que también lleva a la terminación del plan condiciona de casos múltiples, sea cual sea el camino por el cual continúe el programa, al finalizar el del plan condicional de casos múltiples, se continua con el flujo normal del programa en la ilustración siguiente se muestra el plan para la condición múltiple.

**Estructura general del plan condiciona de casos múltiples:**

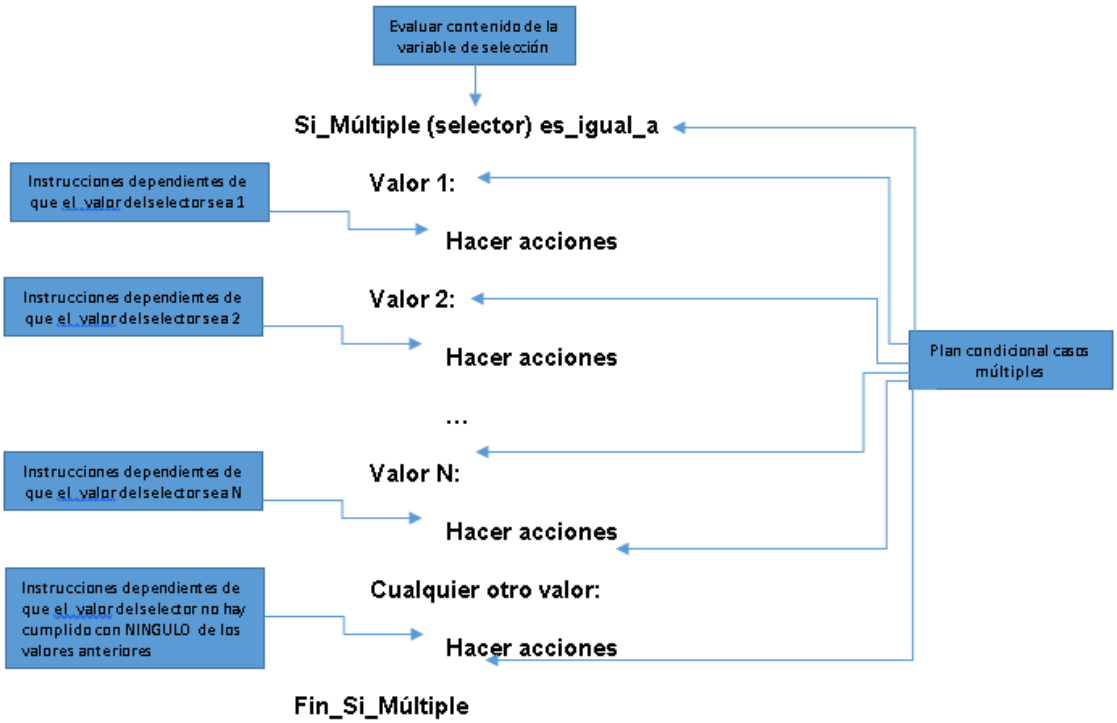



*Ilustración 16: Plan condicional múltiple.*

**Ejemplo 1 de Plan de programación "Condicional MULTIPLE":** El siguiente plan lee el número de día de la semana donde 1=Lunes, 2=Martes… 7=Domingo, en base al número del día leído, deberá de mostrar un mensaje con el nombre correspondiente al número del día introducido.

La ilustración que se muestra a continuación, da solución al problema planteado

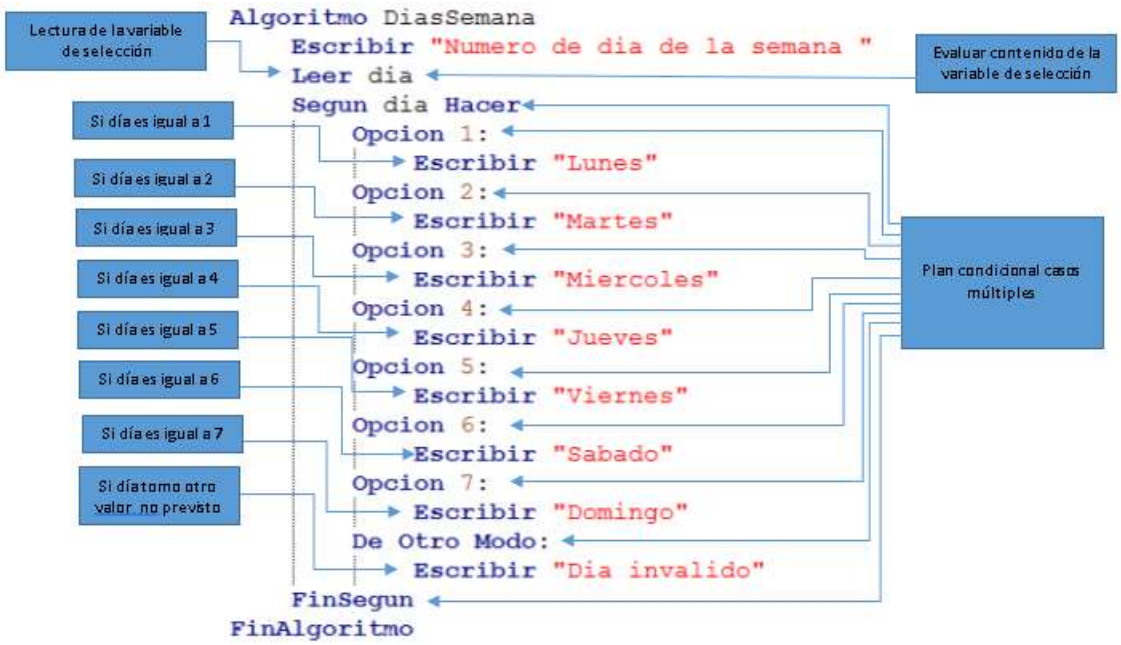



*Ilustración 17: Ejemplo 1 plan condicional múltiple*

En la ilustración 18 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación múltiple.

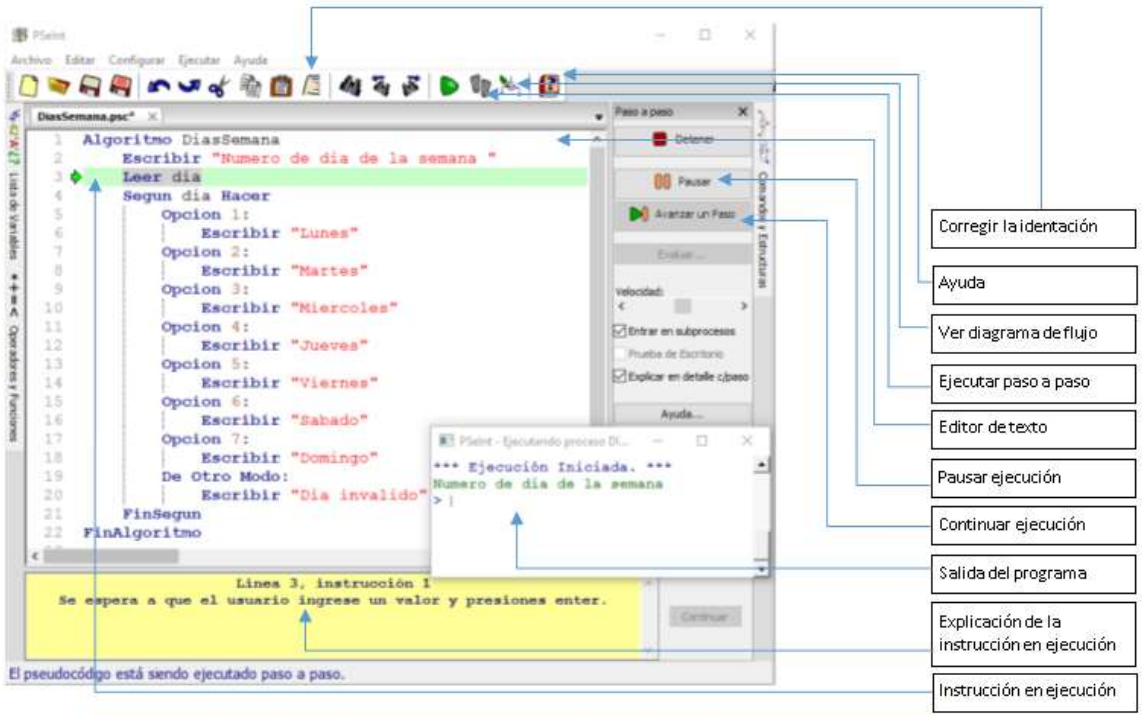



*Ilustración 18: Solución PSeInt ejemplo 1 condicional múltiple*

**Ejemplo 2 de Plan de programación "Condicional MULTIPLE":** El siguiente algoritmo obtiene un valor para una variable llamada VALOR en términos de la evaluación de una función matemática para distintos valores de un selector llamado NÚMERO, la función a evaluar es la siguiente:

| | | |
|---|---|---|
| VALOR= | 100 *DATO | SI NUM = 1 |
| | 100 * DATO ^2 | SI NUM = 2 |
| | 100 / DATO | SI NUM = 3 |
| | 0 | Cualquier otro caso |

La ilustración que se muestra a continuación, da solución al problema planteado

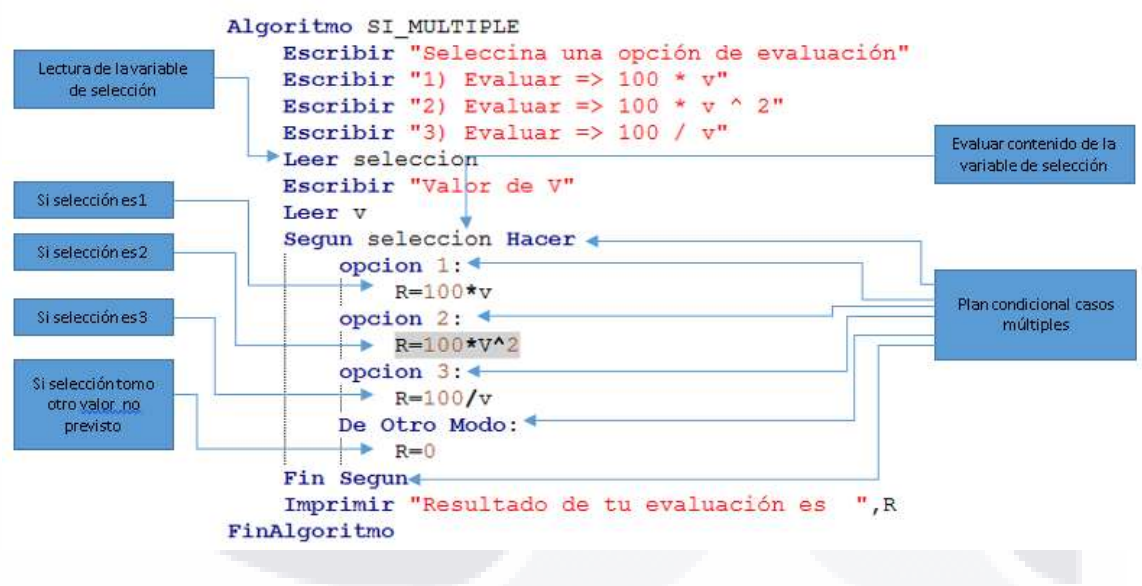



*Ilustración 19: Ejemplo 2 plan condicional múltiple*

En la ilustración 20 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación múltiple.

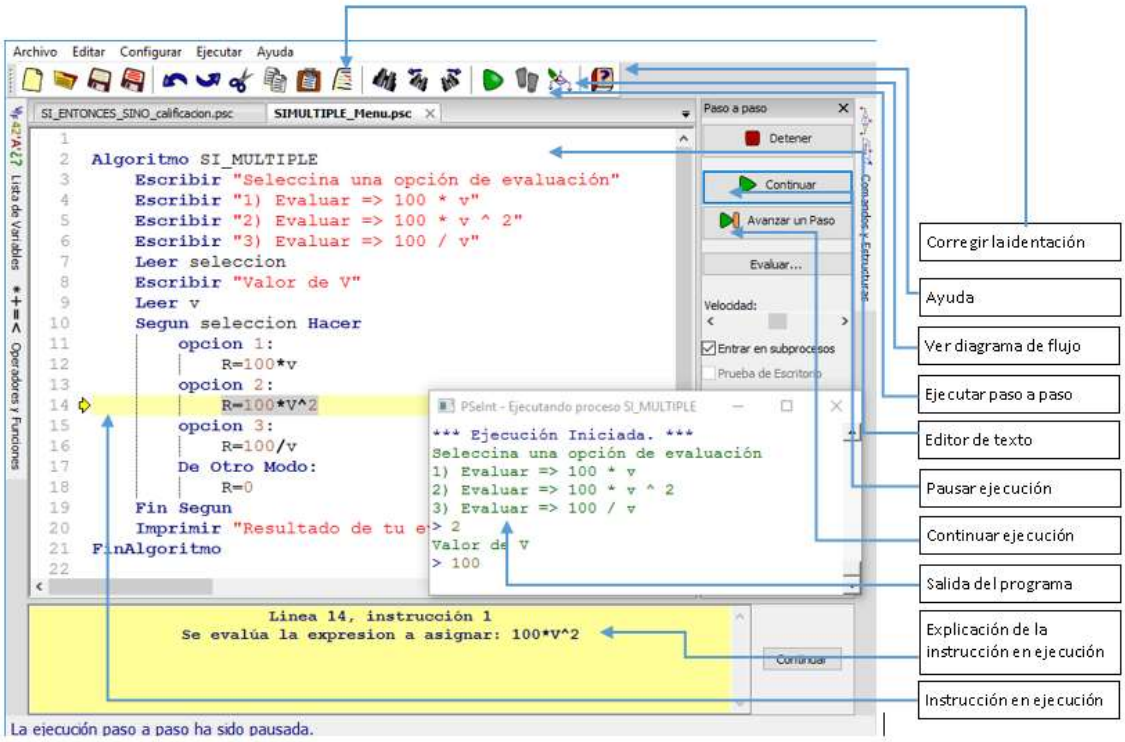



*Ilustración 20: Solución PSeInt ejemplo 1 condicional múltiple*

**Ejercicio 1 de Plan de programación "Condicional MULTIPLE":** El siguiente ejercicio pretende resolver el aumento que se le debe proporcionar a un empleado de X compañía, el aumento se le proporciona de acuerdo a una categoría que se tiene asignada, por lo cual para poder resolverlo es necesario tener la categoría del empleado (CATEGORIA), el sueldo actual (SUELDO) y en base a ello calcular e imprimir el nuevo sueldo, si la categoría no coincide con ninguna, imprimir la leyenda "Categoría invalida", las categorías y aumentos se describen a continuación:

| Categoría | Aumento |
|---|---|
| 1 | 15% |
| 2 | 10% |
| 3 | 8% |
| 4 | 7% |

La ilustración 21 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo SI_MULTIPLE
    Leer CATEGORIA
    Leer  SUELDO
    Segun_____________Hacer
        opcion__:
            AUMENTO = .15
        opcion__:
            AUMENTO = .10
        opcion__:
            AUMENTO = .08
        opcion__:
            AUMENTO = .07
        De Otro Modo:
            Imprimir_____________________
    Fin Segun
    NUEVOSUELDO = SUELDO * AUMENTO + SUELDO
    Imprimir "Tu nuevo sueldo es  " NUEVOSUELDO
FinAlgoritmo
```

*Ilustración 21: Ejercicio 1 por completar condicional múltiple*

### 4.2 Tema: Estructuras de Repetición

En el área de la programación es muy común encontrarse con ejercicios que deben hacer procesos de determinadas operaciones de manera repetitiva. Las instrucciones se repiten una y otra vez mientras que los datos sobre los que trabajan se ven alterados de alguna manera. A este proceso de ejecución de instrucciones realizadas repetidamente se le conoce como CICLO.

Todo CICLO debe terminar con la ejecución después de un determinado número de repeticiones, en cada una de las iteraciones del mismo se deben evaluar condiciones necesarias para saber si el ciclo debe terminar o continuar en su estado de repetición. La condición de evaluación que debe existir en todos los ciclos es la que nos indica cuando debe de terminar el mismo, una condición con respuesta afirmativa, indica que

se debe continuar dentro del ciclo, cuando la condición cambia de verdadero a falso, es cuando se abandona el proceso repetitivo.

Existe 3 tipos de ciclos en la programación, un primer ciclo en el cual está establecido desde el principio el número de veces que este se ha de repetir, a este ciclo se le llamará PARA indicando que se establece el número de repeticiones PARA n veces, existe otro ciclo el cual trabaja en base a una condición, este ciclo debe evaluar la condición a fin de determinar si entra o no a realizar las operaciones repetidas que se tiene programadas dentro del ciclo, si la condición que se evalúa es verdadera, se accede a ciclo y permanece dentro de él mientras dicha condición siga permaneciendo verdadera, a este ciclo se llamará MIENTRAS, y existe un tercer ciclo que hace por lo menos una iteración de las instrucciones, lo primero que hace es ejecutar una vez las instrucciones y al termino evalúa la condición, mientras la condición permanezca verdadera el ciclo continua ejecutando las instrucciones dentro de él, una vez que las condición es falsa, el ciclo termina con su ejecución, a este ciclo se llamará REPETIR.

### 4.2.1 Plan ciclo MIENTRAS

**Descripción:** Esta estructura conocida en los lenguajes de programación como WHILE, es la estructura idónea cuando no se conoce el número de veces que debería repetirse el ciclo, esta estructura tiene una variable que le permite controlar la permanencia dentro del ciclo, mientras se evalúe la condición y esta resulte verdadera, se permanece dentro del ciclo, una vez que la evaluación de la condición resulta falsa, continua con las instrucciones que se encuentren por fuera del ciclo. Es importante tomar en cuenta que la variable de control debe ser modificada dentro del ciclo a fin de que en algún momento la condición que controla el ciclo se pueda evaluar a falso y así terminar el ciclo.

La estructura mientras tiene en general 2 partes:

- La parte cíclica que es el conjunto de instrucciones que se realizan de manera repetida.
- La condición que nos indica cuando se está dentro del ciclo y cuando termina el mismo.

En la ilustración siguiente se muestra la estructura general del plan de repetición MIENTRAS:

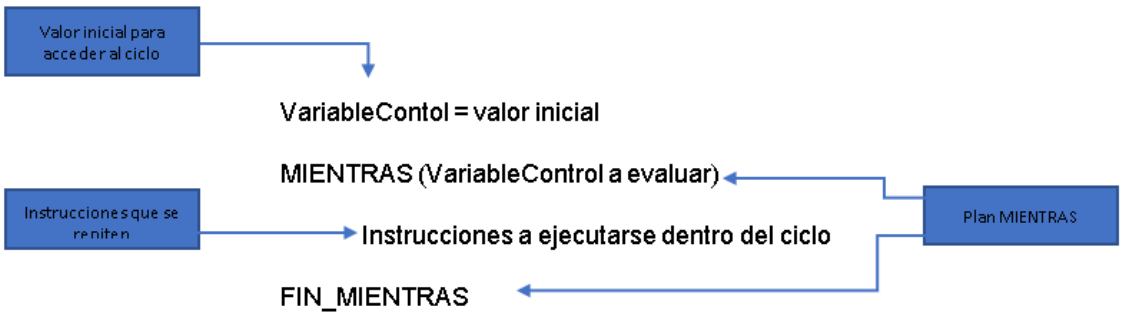



*Ilustración 22: Plan repetición Mientras*

**Ejemplo 1 de Plan de programación "MIENTRAS":** Suponga que quiere contabilizar los gastos que ha hecho durante toda la semana con el objetivo de saber a cuánto asciende su gasto, mientras el gasto no sea cero seguirá sumando los gastos a la cuenta corriente y al final deberá mostrar el total de gastos realizados.

La ilustración que se muestra a continuación, da solución al problema planteado

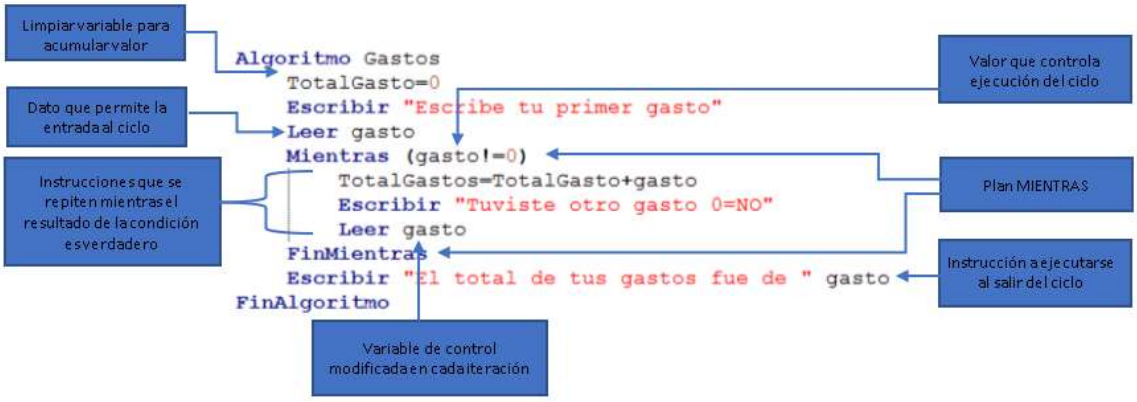



*Ilustración 23: Ejemplo 1 Plan repetición Mientras*

En la ilustración 24 se muestra la solución en el software **PSeInt** al ejemplo 1 del Plan repetición Mientras

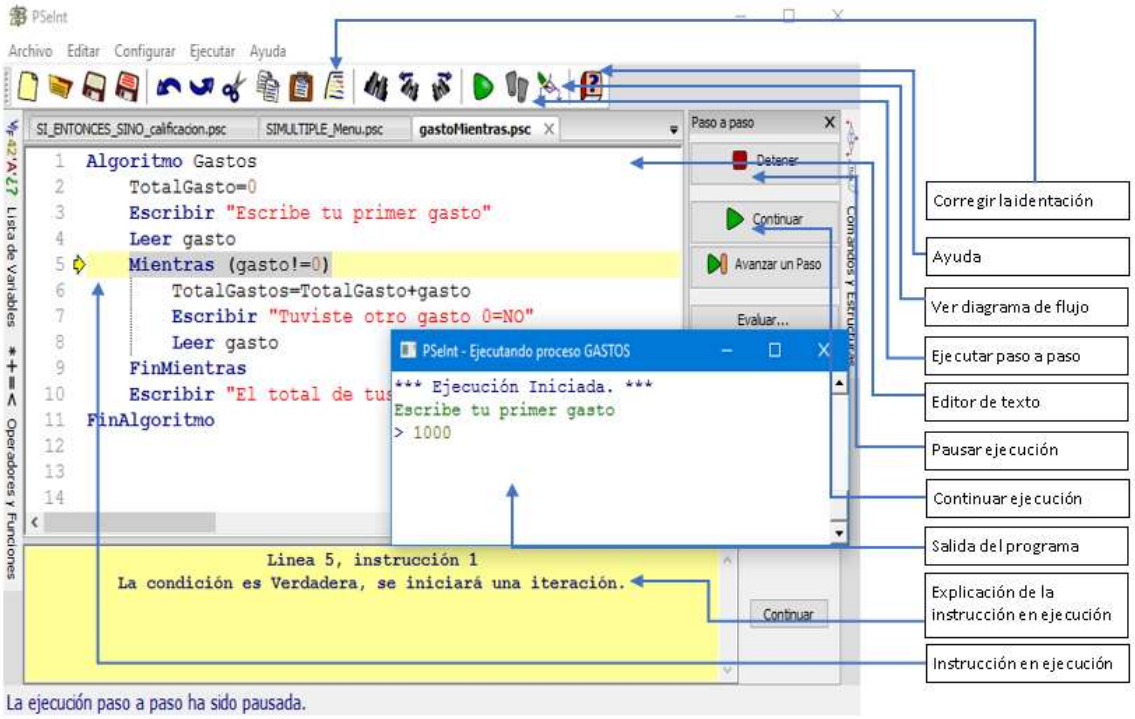



*Ilustración 24: Solución PSeInt ejemplo 1 repetición Mientras*

**Ejemplo 2 de Plan de programación "MIENTRAS":** Suponga que se desean analizar N números enteros, de los cuales se requiere la suma de todos aquellos que sean números pares y el promedio de aquellos que sea impares, el siguiente plan, haciendo uso de los **planes 5 y 8** para lograr la solución al problema planteado.

La ilustración que se muestra a continuación, da solución al problema planteado

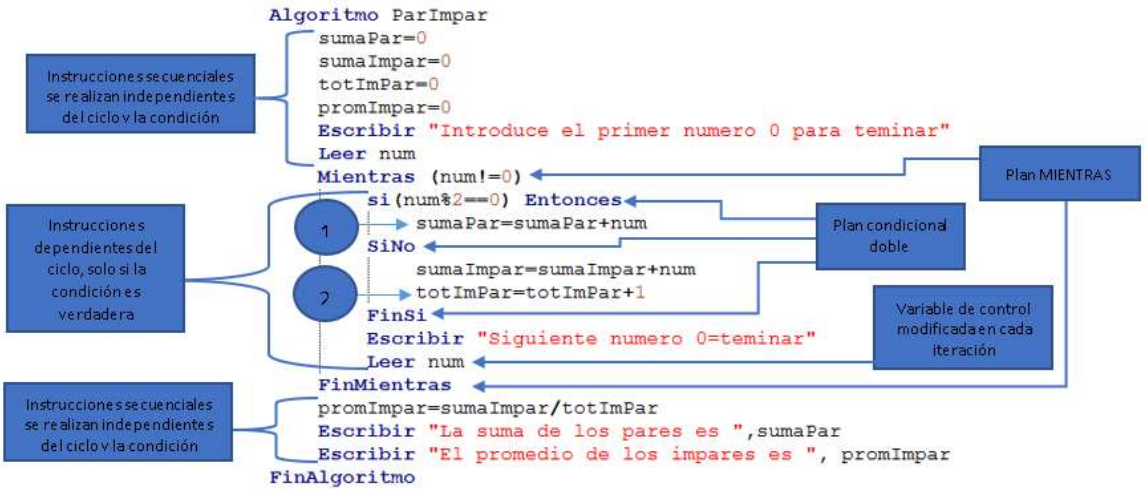



*Ilustración 25: Ejemplo 2 Plan repetición Mientras*

1 Instrucción dependiente de ciclo y posteriormente dependiente de la condición verdadera del condicional doble.

2 Instrucción dependiente de ciclo y posteriormente dependiente de la condición falsa del condicional doble.

En la ilustración 26 se muestra la solución en el software **PSeInt** al ejemplo 2 del Plan repetición Mientras.

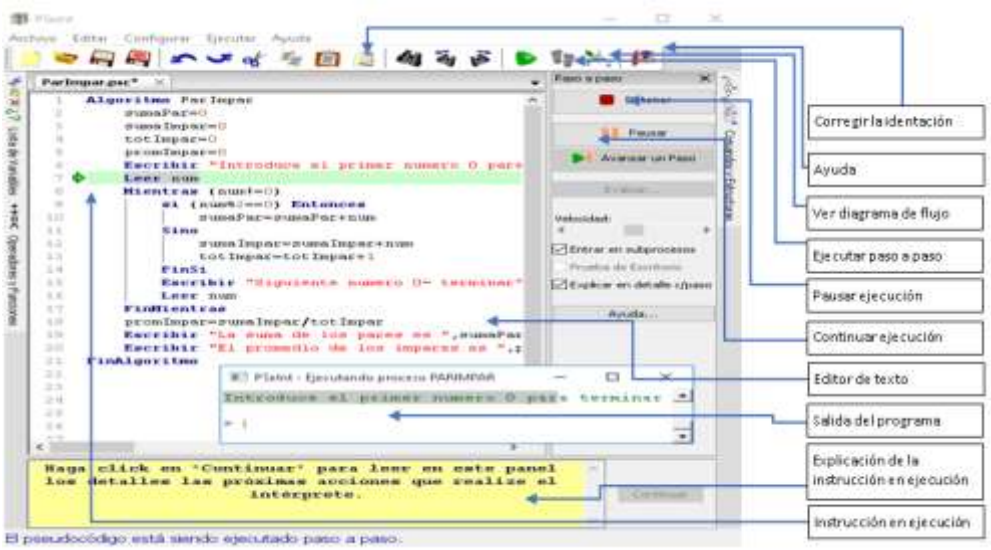



*Ilustración 26: Solución PSeInt ejemplo 2 repetición Mientras*

**Ejercicio1 de Plan de programación "MIENTRAS":** Se requiere hacer un algoritmo que pida a un usuario una serie de números cualquiera, que los sume y vaya visualizado la suma en la pantalla y que deje de leerlo cuando el usuario ingrese un número mayor a 100.

La ilustración 27 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo AnalisisNumeros
    suma=0
    Escribir "Introduce un numero entero, un nuemero mayor a 100 indica el fin de datos"
    Leer num;
    Mientras (_____________) Hacer
        suma=suma+num
        Escribir "La sumatoria de los numeros introducidos es ",suma
        Escribir "Introduce un numero entero, un nuemero mayor a 100 indica el fin de datos"
        Leer _________
    FinMientras
FinAlgoritmo
```

*Ilustración 27: Ejercicio 1 por completar repetición Mientras*

**Ejercicio 2 de Plan de programación combinado "MIENTRAS" y "Condicional Simple":** Se requiere un algoritmo que dados N números enteros como datos, realice lo siguiente:

a) Obtenga cuántos números leídos fueron mayores que cero.

b) Calcule el promedio de los números positivos.

c) Obtenga el promedio de todos los números.

La lectura de datos termina cuando el usuario introduzca un 0 como dato.

La ilustración 28 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo AnalisisNumeros
    mayoresCeroCero=0
    sumaPositivos=0
    promedioPositivos=0
    promedioGeneral=0
    totalNumeros=0
    sumaTotal=0
    Escribir "Introduce un numero entero 0=Fin de datos"
    Leer num;
    Mientras (_____________) Hacer
        Si(__________ ) Entonces
            mayoresCero=mayoresCero+1
            sumaPositivos=sumaPositivos+num
        FinSi
        totalNumeros=totalNumeros+1
        sumaTotal=sumaTotal+num
        Escribir "Introduce un numero entero 0=Fin de datos"
        Leer_____________
    FinMientras
    promedioPositivos=sumaPositivos/mayoresCero
    promedioGeneral=sumaTotal/totalNumeros
    Escribir "Numero mayores que cero son ",mayoresCero
    Escribir "Promedio de los numero positivos es ",promedioPositivos
    Escribir "Promedio de todos los numeros es ",promedioGeneral
FinAlgoritmo
```

*Ilustración 28: Ejercicio 2 por completar repetición Mientras*

### 4.2.2 Plan ciclo REPETIR

**Descripción:** Esta estructura conocida en la programación como do-while, es otra solución cuando no se conoce el número de veces que debe de repetirse el ciclo, a diferencia de la estructura MIENTRAS (while), este tipo de ciclo siempre hace por lo menos una iteración del mismo, ya que la condición de control se encuentra al final del ciclo, esto provoca que se ejecuten las instrucciones dentro del ciclo por lo menos una vez y posteriormente se evalúe la condición de permanencia dentro del ciclo, una vez evaluada la condición, si esta resulta verdadera, se permanece dentro del ciclo un número definido de veces mientras la condición así lo permita, una vez que la condición se evalúa a falso, se sale del ciclo y se continua con las instrucciones que se encuentren fuera del mismo. Al igual que el ciclo MIENTRAS, cuenta con una variable de control que debe ser modificada dentro del ciclo, recuerde que, si esta condición no se modifica, entonces se entra en un ciclo infinito.

En general dentro de la estructura repetir se pueden distinguir 2 partes:

- La parte cíclica: que es el conjunto de instrucciones que se realizan de manera repetida.
- La condición: que marca la permanencia dentro del ciclo

En la ilustración siguiente se muestra la estructura general del plan de repetición Repetir:

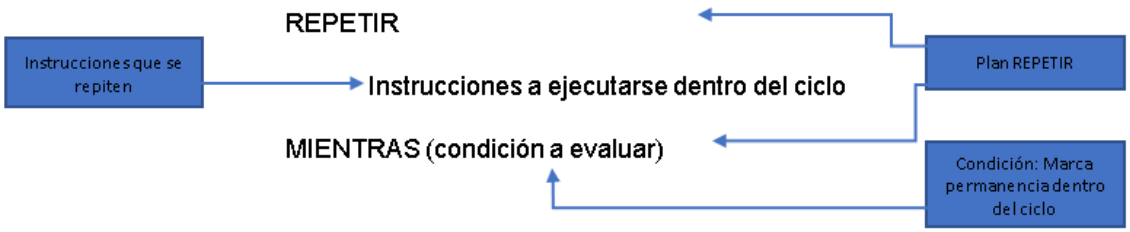



*Ilustración 29: plan de repetición Repetir*

**Ejemplo 1 de Plan de programación "REPETIR":** Se requiere un algoritmo que sume N números, cuando se introduzca un 0 se termina la lectura de datos y se muestra la suma de todos los números introducidos.

La ilustración que se muestra a continuación, da solución al problema planteado

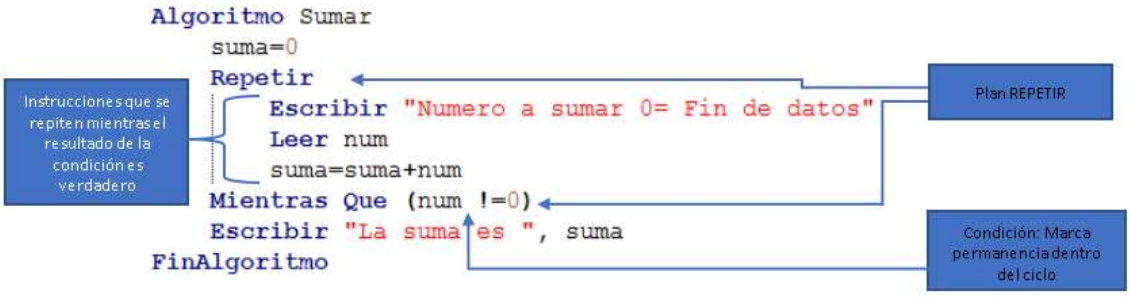



*Ilustración 30: Ejemplo 1 plan Repetir*

En la ilustración 31 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación Repetir.

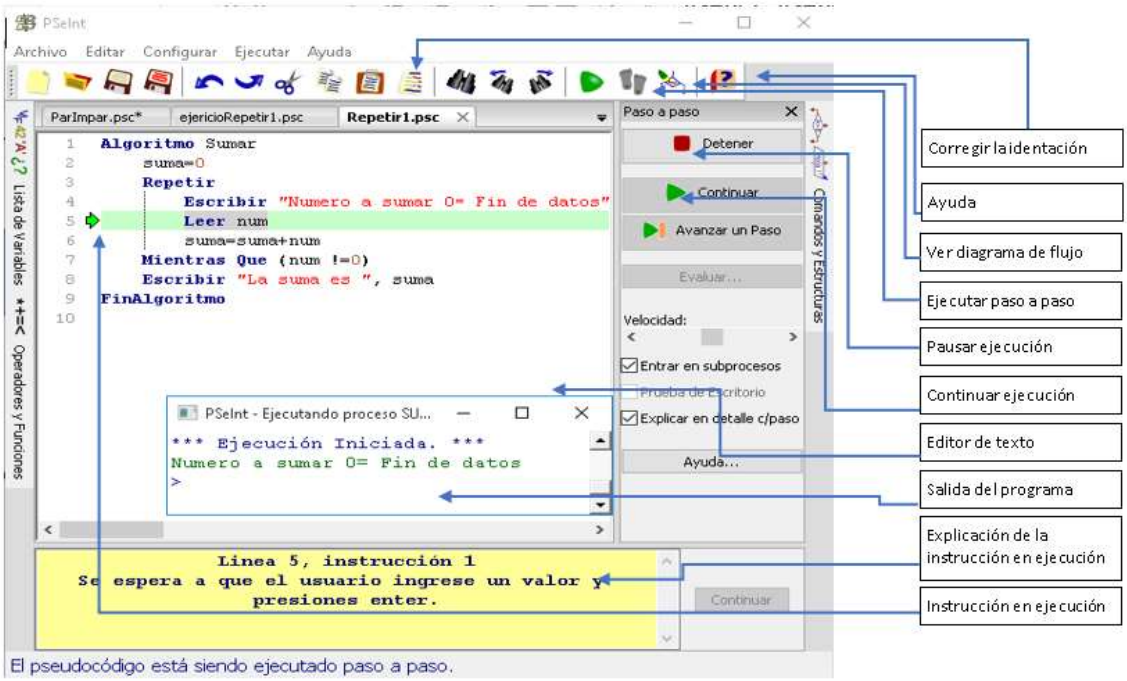



*Ilustración 31: Solución PSeInt ejemplo 1 plan Repetir*

**Ejemplo 2 de Plan de programación "REPETIR":** Construya un algoritmo que lea N números, calcule el mayor y el menor de estos números. La marca de fin de datos está definida por un 0 en la lectura de los mismos.

La ilustración que se muestra a continuación, da solución al problema planteado.

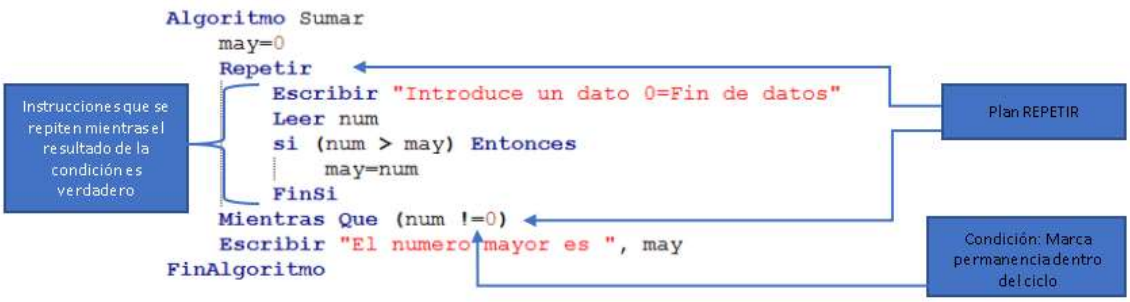



*Ilustración 32: Ejemplo 2 plan Repetir*

En la ilustración 33 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación Repetir.

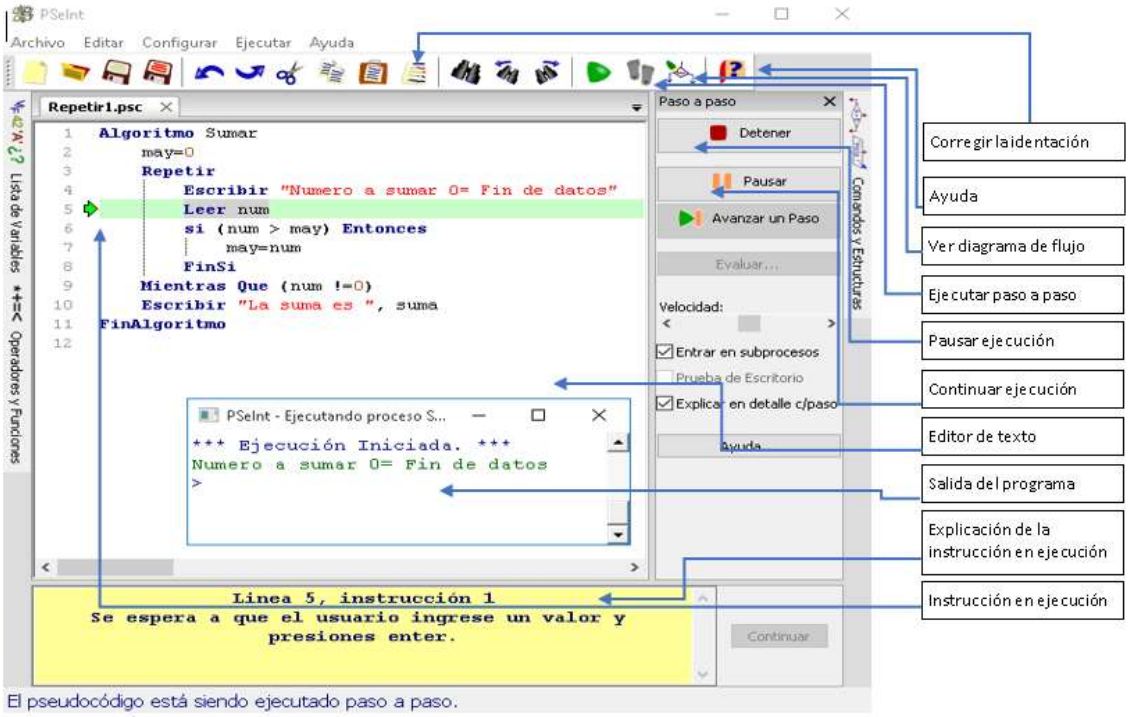



*Ilustración 33: Solución PSeInt ejemplo 2 repetición Mientras*

**Ejemplo 3 de Plan de programación "REPETIR":** Un vendedor ha hecho una serie de ventas y desea conocer aquellas de $200 o menos, las mayores a $200 pero inferiores a $400, y el número de ventas de $400 o superiores a tal cantidad, así como el total de las ventas realizadas. Haga un algoritmo que le proporcione al vendedor esta información después de haber leído los datos de cada una de las ventas realizadas.

La ilustración que se muestra a continuación, da solución al problema planteado

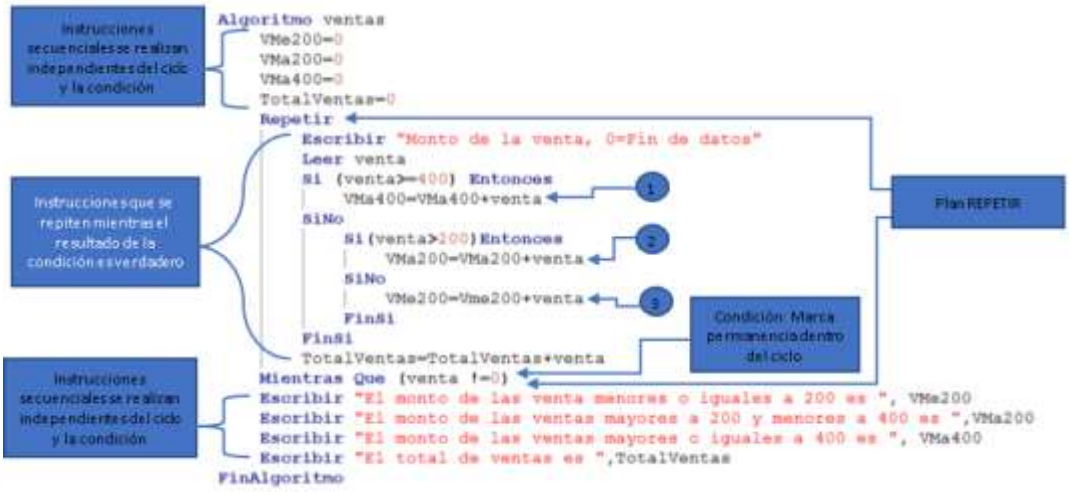



*Ilustración 34: Ejemplo 3 plan Repetir*

1. Instrucciones dependientes del ciclo y de la parte verdadera del condicional doble
2. Instrucciones dependientes del ciclo, del sino del condicional principal y de la parte verdadera del condicional doble dentro del sino del condicional principal
3. Instrucciones dependientes del ciclo, del sino del condicional secundario y de la parte falsa del condicional doble dentro del sino del condicional principal

En la ilustración 35 se muestra la solución en el software **PSeInt** al ejemplo 3 del plan de programación Repetir.

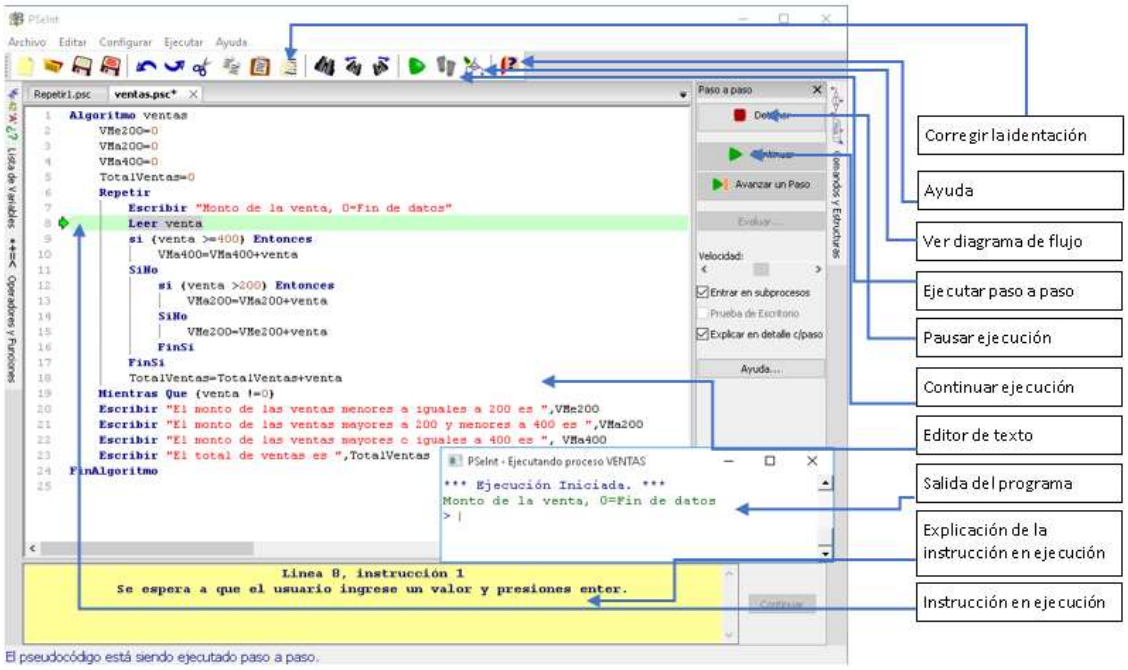



*Ilustración 35: Solución PSeInt ejemplo 3 repetición Repetir*

**Ejercicio1 de Plan de programación "REPETIR":** Suponga que requiere un algoritmo que valide si una clave introducida por el usuario es válida, para que la clave leída sea válida, debe contener el valor 1234, de lo contrario el programa seguirá pidiendo una clave hasta llegar a obtener la clave que se está verificando.

La ilustración 36 muestra una solución parcial al problema planteado, escriba en la línea la instrucción que falta para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo NIP
    Repetir
        Escribir "Escribe tu NIP "
        Leer clave
    Mientras Que (_____________________)
FinAlgoritmo
```

*Ilustración 36: Ejercicio 1 por completar Repetir*

**Ejercicio2 de Plan de programación "REPETIR":** Supónga que en una reciente elección hubo cuatro candidatos (con identificadores 1, 2, 3, 4). Usted habrá de encontrar, mediante un algoritmo, el número de votos correspondiente a cada candidato y el porcentaje que obtuvo respecto al total de los votantes. El usuario tecleará los votos de manera desorganizada, tal y como se obtuvieron en la elección, el final de datos está representado por un cero.

Observe, como ejemplo, la siguiente lista: 1 3 1 4 2 1 4 1 1 1 2 1 3 1 4 0
Donde 1 representa un voto para el candidato 1; 3 un voto para el candidato 3; y así sucesivamente.

La ilustración 37 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo candidatos
    VtosC1 = 0
    VtosC2 = 0
    VtosC3 = 0
    VtosC4 = 0
    TotalVotos=0
    Repetir
        Escribir 'Vota para candidato '
        Leer__________
        Segun voto Hacer
            1:VtosC1=_______+1
            2:VtosC2=VtosC2+1
            3:__________________
            4:VtosC4=VtosC4+___
        FinSegun
    Mientras Que (__________________)
    TotalVotos=VtosC1+VtosC2+VtosC3+VtosC4
    PctjeC1=VtosC1/TotalVotos*100
    PctjeC2=VtosC2/TotalVotos*100
    PctjeC3=VtosC3/TotalVotos*100
    PctjeC4=VtosC4/TotalVotos*100
    Escribir "Votos cadidato 1 ",_________," Equivalente al ",_________,"%"
    Escribir "Votos cadidato 2 ",_________," Equivalente al ",_________,"%"
    Escribir "Votos cadidato 3 ",_________," Equivalente al ",_________,"%"
    Escribir "Votos cadidato 4 ",_________," Equivalente al ",_________,"%"
FinAlgoritmo
```

*Ilustración 37: Ejercicio 1 por completar Repetir.*

### 4.2.3 Plan ciclo PARA

**Descripción:** Esta estructura conocida en los lenguajes como ciclo FOR, es ideal cuando se conoce con antelación el número de veces que quiere que se ejecute el ciclo, al igual que los dos casos anteriores tiene una variable de control que como su nombre lo dice controla el número de veces que se ejecutan las instrucciones dentro del ciclo, dentro del ciclo PARA se pueden distinguir 3 partes esenciales:

- VARIABLE INICIO: marca el inicio del ciclo
- FIN: indica el valor hasta donde se ejecuta el ciclo
- TAMAÑO DEL SALTO: indica de cuanto en cuanto ira avanzando la variable de control hasta que llegue a su valor final

**En la ilustración siguiente se muestra la estructura general del plan de repetición PARA:**


*Ilustración 38: Plan de repetición PARA*

**Ejemplo 1 de Plan de programación "PARA":** Se necesita un algoritmo que imprima por pantalla los números del 1 al 1000, estos deberán ir mostrándose de 2 en 2

La ilustración que se muestra a continuación, da solución al problema planteado.


*Ilustración 39: Ejemplo 1 plan condicional Para*

En la ilustración 40 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación **Para**.

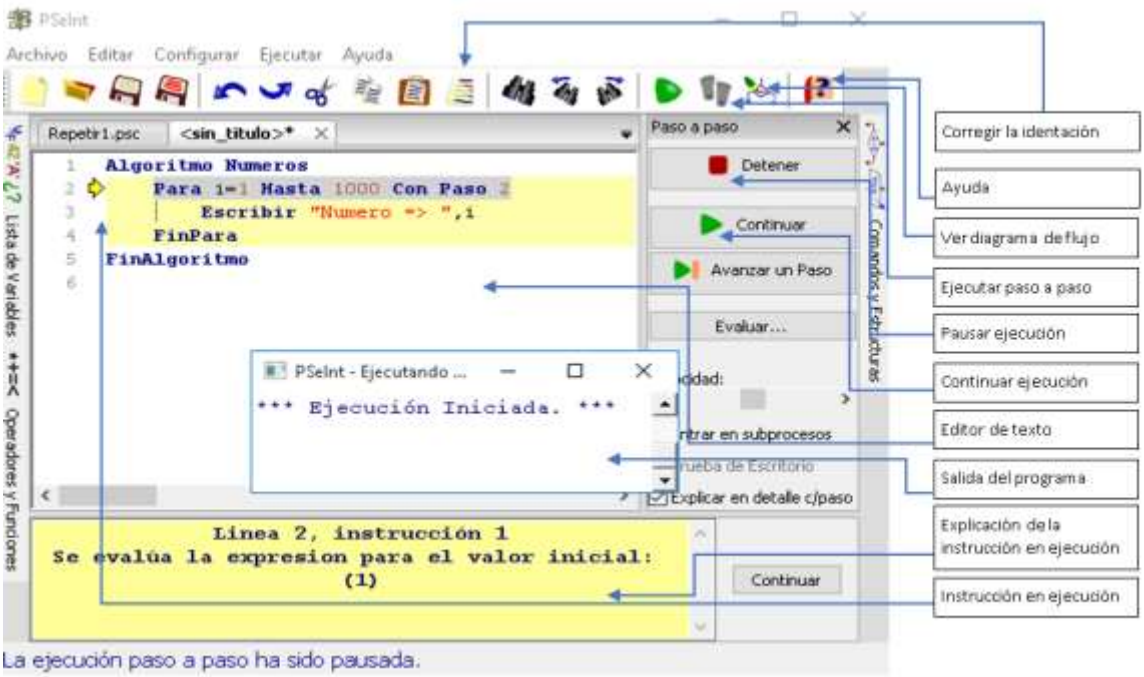



*Ilustración 40: Ejemplo 1 plan repetición Para*

**Ejemplo 2 de Plan de programación "PARA":** Se requiere un algoritmo que reciba un número del usuario y obtenga el factorial de dicho número, recuerde que el factorial de un número se obtiene multiplicando el número por todos los anteriores hasta llegar a 1, por ejemplo el factorial de 6 es 6 x 5 x 4 x 3 x 2 x 1 = 720.

La ilustración que se muestra a continuación, da solución al problema planteado

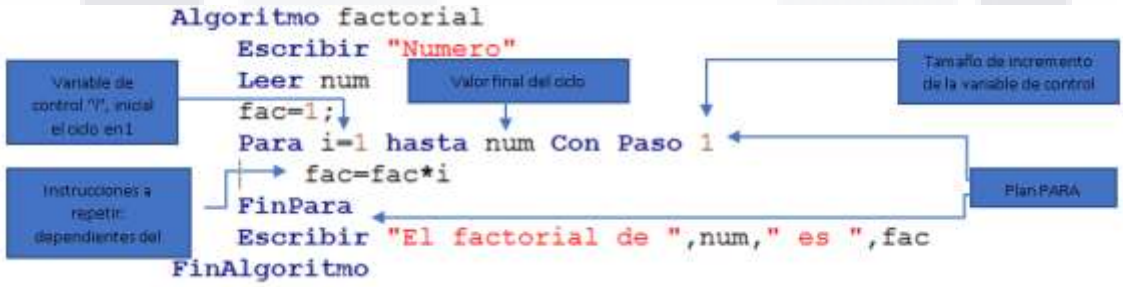



*Ilustración 41: Ejemplo 2 plan repetición Para*

En la ilustración 42 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación Para.

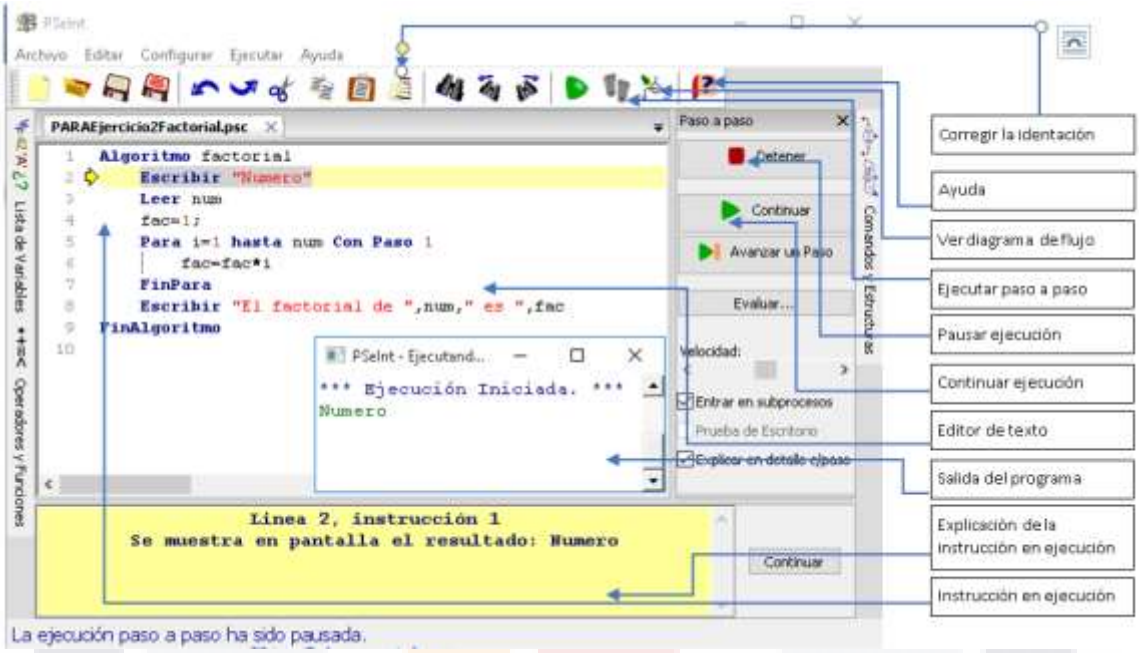



*Ilustración 42: Solución PSeInt ejemplo 2 repetición Para*

**Ejemplo 3 de Plan de programación "PARA":** Se requiere un algoritmo que saque la suma de los números pares y el promedio de los números impares contenidos entre dos valores A y B dados por el usuario.

La ilustración que se muestra a continuación, da solución al problema planteado

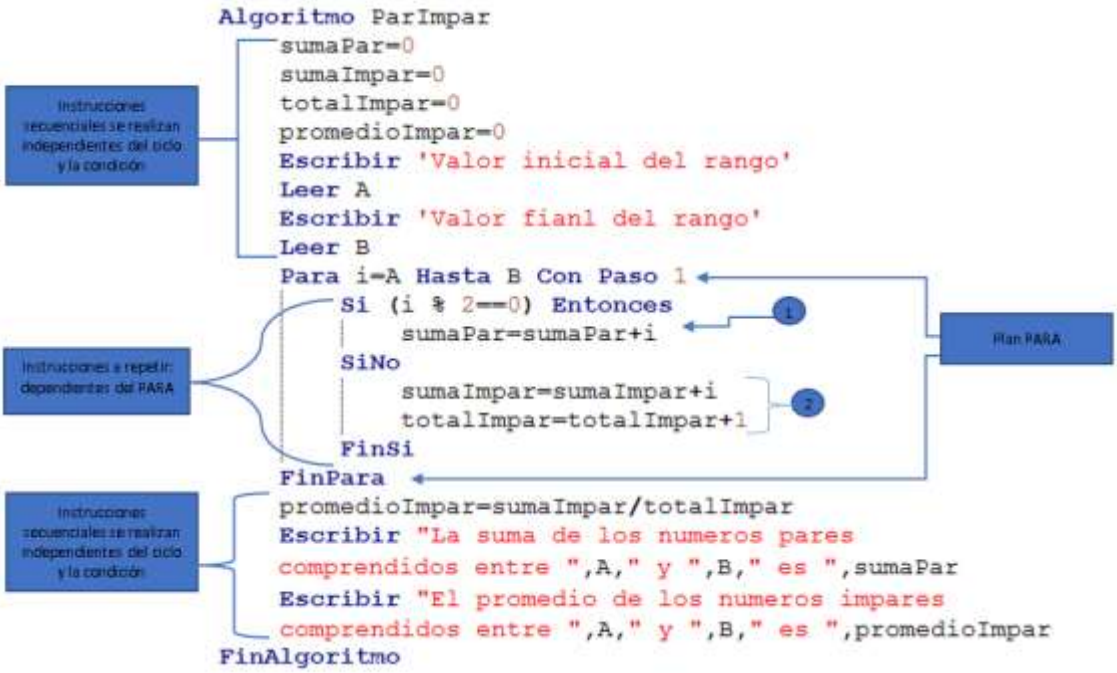



*Ilustración 43: Ejemplo 3 plan repetición Para*

1. Instrucción dependiente del ciclo (PARA) y posteriormente de la condición verdadera del condicional doble
2. Instrucción dependiente del ciclo (PARA) y posteriormente de la condición falsa del condicional doble

En la ilustración 44 se muestra la solución en el software **PSeInt** al ejemplo 3 del plan de programación repetición Para.

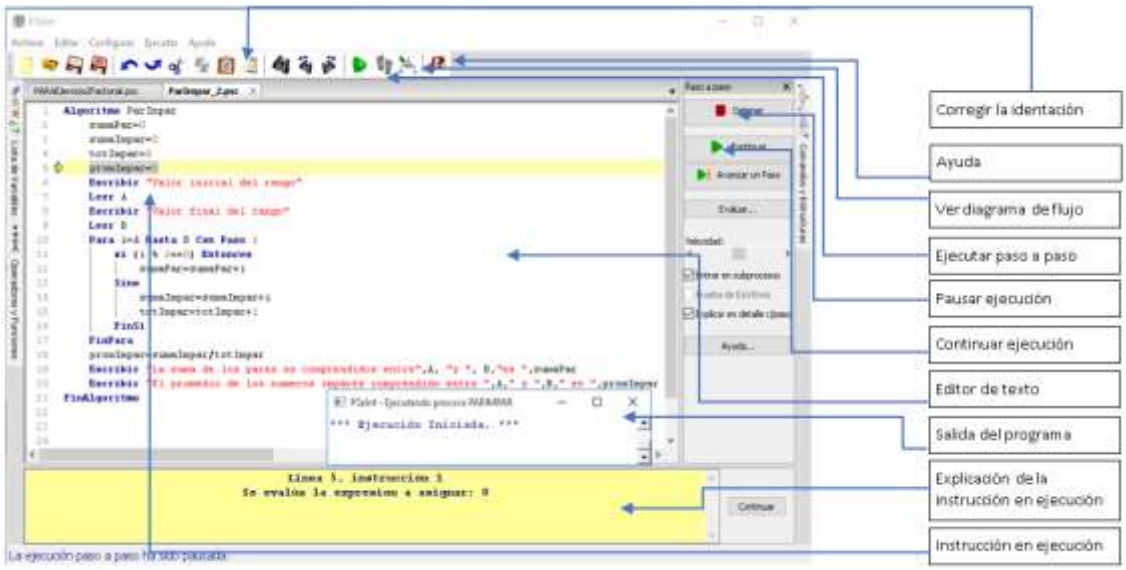



*Ilustración 44: Solución PSeInt ejemplo 3 repetición Para*

**Ejercicio1 de Plan de programación "PARA":** La serie Fibonacci, es una secuencia de números que descubrió el matemático Leonardo de Pisa, esta serie comienza con los números 0 y 1, el siguiente número en la serie se obtiene como la sumatoria de los dos anteriores, de tal manera que, si se requieren 8 números de la serie Fibonacci, el resultado sería: 0-1-1-2-3-5-8-13, el siguiente algoritmo intenta resolver esta serie imprimiendo una cantidad de números que es determinada por el usuario.

La ilustración 45 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo Fibonacci
    Escribir "cantidad de numeros de la secuencia"
    Leer __________
    fib1= ______
    fib2= ______
    Escribir Sin Saltar "0-1"
    Para i=__ hasta_____ Con Paso____
        fib=_________________
        Escribir Sin Saltar "-",fib
        fib1=fib2
        fib2=fib
    FinPara
FinAlgoritmo
```

*Ilustración 45: Ejercicio 1 por completar repetición Para*

**Ejercicio 2 de Plan de programación "PARA":** Se requiere un algoritmo que imprima las tablas de multiplicar desde la del 1 a la del 10 en el formato 1 x 1 = 1, 1 x 2 = 2,…, 10 x 10 = 100, el siguiente algoritmo intenta imprimir estas tablas.

La ilustración 46 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo TablasMultiplicar
    Para i=__ hasta_____ Con Paso____
        Para __=1 hasta___ Con Paso 1
            Escribir ___,"X",___,"=",i*j
        FinPara
    FinPara
FinAlgoritmo
```

*Ilustración 46: Ejercicio 2 por completar repetición Para*

### 4.3 Tema: Contadores y acumuladores

**Descripción:**

En programación, se llama **contador** a una variable cuyo valor se incrementa o decrementa en un valor fijo (en cada iteración de un ciclo). Un contador suele utilizarse

para contar el número de veces que itera un bucle. Pero, a veces, se utiliza para contar, solamente, aquellas iteraciones de un bucle en las que se cumpla una determinada condición.

Requisitos de conocimiento previos: Es necesario que se conozcan los tipos de datos, la declaración de variables.

Casos típicos de uso: Se usan dentro de estructuras repetitivas como pueden ser ciclos para (for), mientras (while) o repetir (do-while), siempre que se quiere contabilizar algo, la forma de hacerlo es mediante un acumulador o un contador.

En la ilustración siguiente se muestra la estructura general del plan CONTADOR/ACUMULADOR:

### 4.3.1 Plan CONTADOR / ACUMULADOR

**Estructura:**


*Ilustración 47: Plan Contador/Acumulador*

**Ejemplo 1 de Plan de programación "Contadores":** Se requiere un contador que vaya contando (acumulando) de 1 en 1, la estructura básica del mismo se muestra en la siguiente ilustración:


*Ilustración 48: Ejemplo 1 plan Contador/Acumulador*

**Ejemplo 2 de Plan de programación "Contadores"**: Se requiere mostrar en pantalla la sumatoria de los valores de 2 en 2, hasta obtener el valor de 200.

La ilustración que se muestra a continuación, da solución al problema planteado

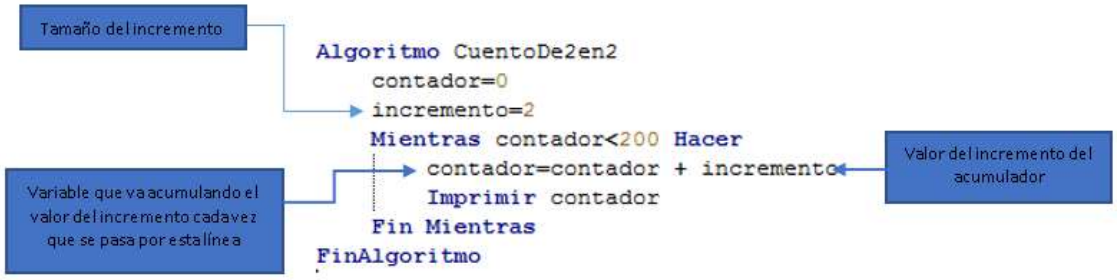



*Ilustración 49: Ejemplo 2 plan Contador/Acumulador*

En la ilustración 50 se muestra la solución en el software **PSeInt** al ejemplo 2 del plan de programación Contador/Acumulador.

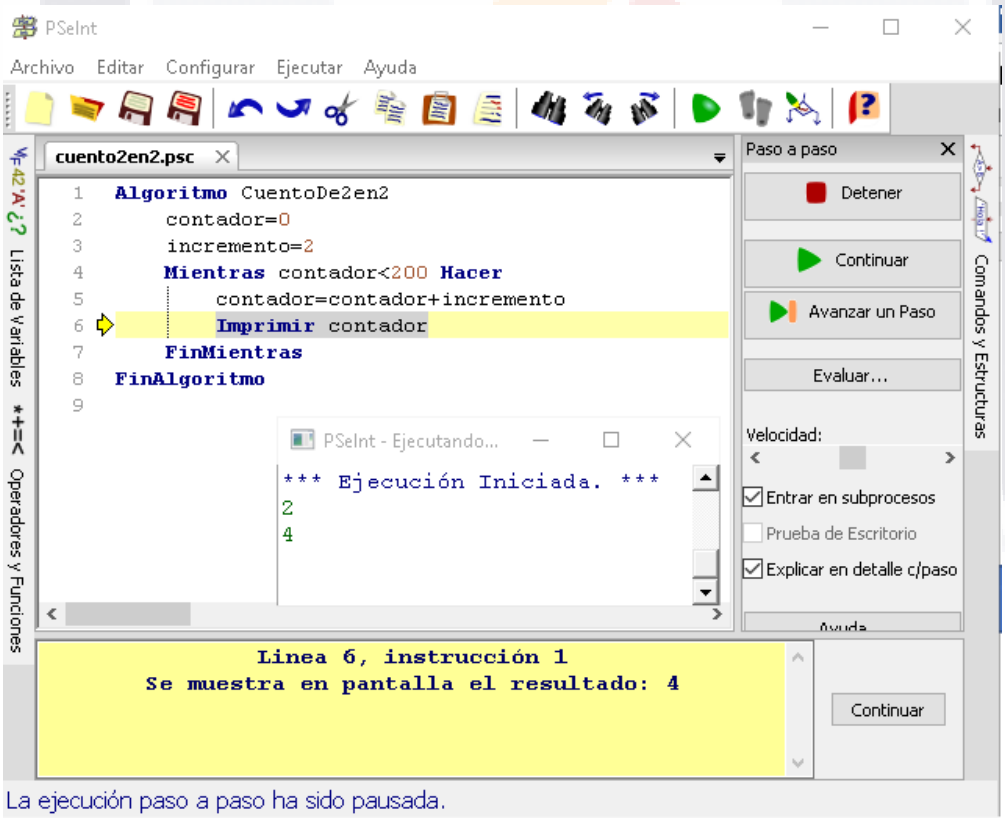



*Ilustración 50: Solución PSeInt ejemplo 2 plan Contador/Acumulador*

**Ejemplo 3 de Plan de programación "Contadores"**: Ahora, se requiere mostrar en pantalla la sumatoria de los valores de 5 en 5, comenzando en 5 y hasta obtener el valor de 500.

La ilustración que se muestra a continuación, da solución al problema planteado

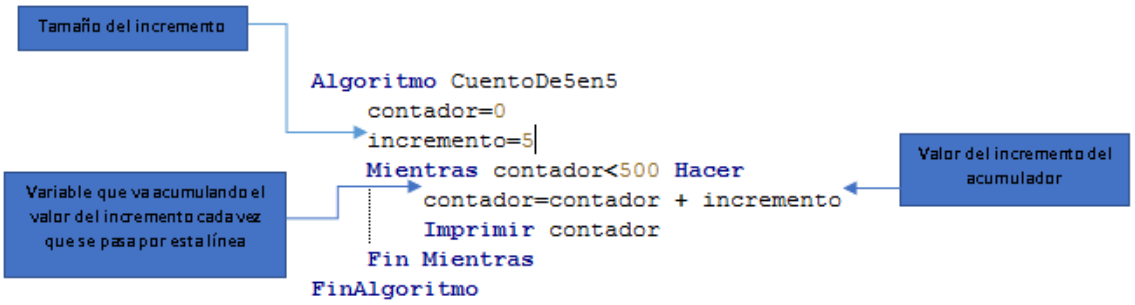



*Ilustración 51: Ejemplo 3 plan Contador/Acumulador*

En la ilustración 52 se muestra la solución en el software **PSeInt** al ejemplo 3 del plan de programación Contador/Acumulador.

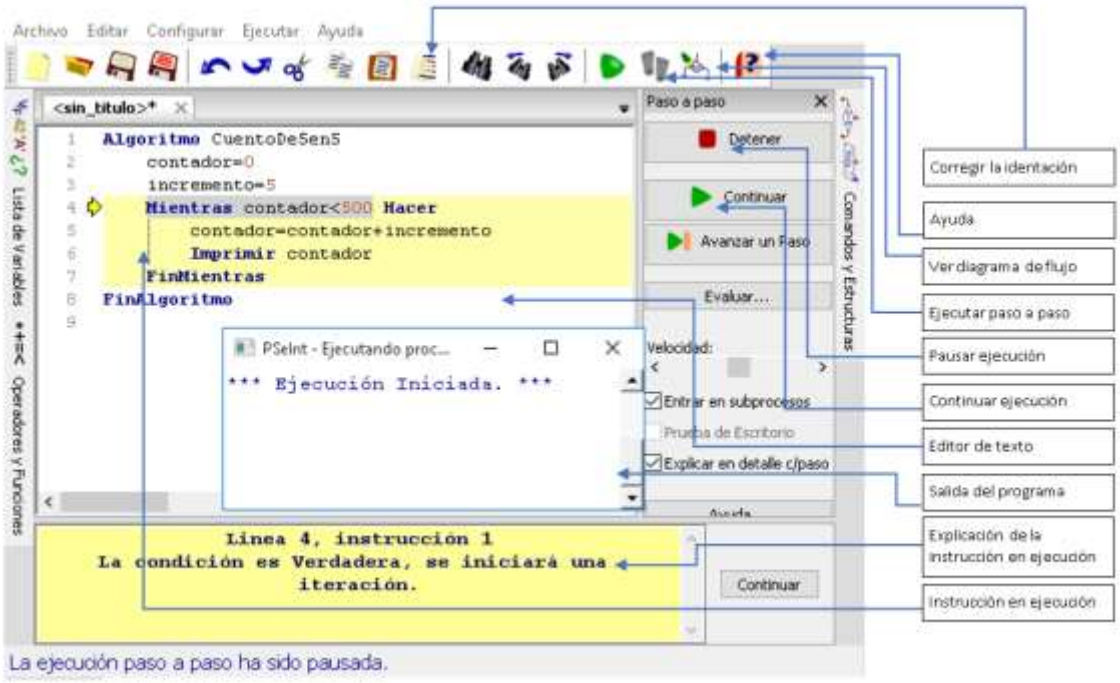



*Ilustración 52: Solución PSeInt ejemplo 3 plan Contador/Acumulador*

**Ejercicios:** El siguiente código pretende hacer una cuenta de datos, iniciando en el valor 20 y terminando en el valor 2000, el conteo debe hacer incrementos de 10 en 10.

La ilustración 53 muestra una solución parcial al problema planteado, escriba en las líneas las instrucciones que faltan para logar que se realice de manera **correcta** la solución a dicho problema.

```
Algoritmo CONTADORES
    CONTADOR =_____
    INCREMENTO =_____
    Mientras CONTADOR <_____ Hacer
        _____________________
    Fin Mientras
    Imprimir CONTADOR
FinAlgoritmo
```

*Ilustración 53: Ejercicio 1 por completar Contador/Acumulador*

### 4.4 Plan Arreglos/Vectores

**Descripción:**

Un arreglo se define como una colección finita, homogénea y ordenada de elementos (Dr. Osvaldo Cairo Battistutti, 2005, p. 181).

Finita: todo arreglo tiene un límite, es decir se debe determinar cuál será el número máximo de elementos que podrán formar parte del arreglo.

Homogénea: todos los datos de un arreglo son del mismo tipo (enteros, reales, caracteres, etc.,).

Ordenada: Siempre puedo conocer la posición de cualquier elemento, es decir, se cuál es el elemento uno, dos, tres, hasta el n-esimo.

**Observaciones:**

a) El índice del arreglo puede ser cualquier tipo ordinal (carácter, entero, etc.).
b) Los elementos que se almacenan dentro de él, pueden ser de cualquier tipo (entero, real, cadena de caracteres, registro, arreglo, etc.).

c) Generalmente se usan “[ ]” para indicar la posición de los elementos dentro del arreglo, es decir el índice del mismo. Dentro de los [ ], se escribe un valor ordinal (puede ser una variable, una constante o una expresión tan compleja, siempre y cuando el valor final obtenido de la misma se un ordinal).

**Requisitos de conocimiento previos:** Es necesario que se conozcan los tipos de datos, la declaración de variables, los ciclos, forma de entrada de los datos y formas de impresión de los mismos.

**Casos típicos de uso:** Siempre que se requiere almacenar un conjunto de datos con las características antes definidas, se pude usar un arreglo para almacenarlos, pude ser un conjunto de calificaciones de alumnos, un conjunto de nombres de personas, un conjunto de promedios de alumnos, etc., los arreglos tienen múltiples y variados usos.

**Declaración de arreglos.**

Antes de proceder al uso de un arreglo para almacenar elementos, lo primero que se debe hacer es declarar el mismo, es decir indicarle al programa que va a hacer uso de una variable tipo arreglo para almacenar información, recuerde que los arreglos tienen tres características, son finitos, ordenados y homogéneos, siguiendo estas características, una declaración quedaría como se ilustra a continuación:

### 4.4.2 Estructura del plan Arreglos/Vectores:


*Ilustración 54: Plan Vector*

**Donde:**

Tipo: indica el tipo de datos de cada uno de los elementos que almacena el arreglo, con esto se cumple con la característica de homogeneidad de los mismos, todos los elementos almacenados serán del tipo que se defina.

NombreArreglo: Es el nombre con el cual se va a usar el arreglo a lo largo de todo su programa.

Tamaño del arreglo: es un valor entero que indica el número de elementos que almacena un arreglo.

**Ejemplo 1 de Plan de programación "Arreglos/Vectores":** Se requiere definir un vector para almacenar 100 edades de personan, las edades de las personas se dan en números enteros, debido a ello la declaración seria como sigue:

La ilustración que se muestra a continuación, da solución al problema planteado


*Ilustración 55: Ejemplo 1 plan Vector*

## Ejemplo del espacio en memoria usado:

| Edad 1 | Edad 2 | Edad 3 | Edad 4 | Edad 5 | Edad 6 | Edad 7 | Edad 8 | ……. | Edad 100 |
|---|---|---|---|---|---|---|---|---|---|

**Ejemplo 2:** Declarar un vector para almacenar 50 promedios de alumnos, ordinariamente los promedios de los alumnos se dan en números con parte decimal, de ahí que el tipo de estos debería ser un tipo que nos permita almacenar valores con punto flotante.

Flotantes promedios [50]

## Ejemplo del espacio en memoria usado:

| Promedio 1 | Promedio 2 | Promedio 3 | Promedio 4 | …… | Promedio 50 |
|---|---|---|---|---|---|

**Las operaciones básicas a realizar en arreglos son:**

- Lectura y/o escritura de datos
- Asignación de datos
- Actualización
    - Agregar datos a los ya existentes
    - Eliminación de datos ya existentes
    - Modificación de datos ya existentes
- Buscar datos
- Ordenar los datos

**Lectura de datos**

Los arreglos son un conjunto de datos agrupados bajo un mismo nombre, por lo cual también se les conoce como datos estructurados, debido a ello la operación de lectura de datos no se pude hacer de una sola vez o en una sola línea de código para todo el arreglo, para ello es necesario que se haga elemento a elemento del arreglo.

El proceso de leer los datos de un arreglo, es básicamente llenar cada uno de sus elementos con un algún valor específico, donde el valor generalmente está dado por el usuario.

Debido a que un arreglo es un número finito de elementos, la manera de leer los datos más práctica y más sencilla suele ser por medio de un ciclo que recorra todas las posiciones del arreglo y vaya llenado cada una de ellas conforme las recorre, así pues, el proceso de llenado de un arreglo es generalmente el siguiente:

- Iniciar con un ciclo que vaya desde la posición 1 hasta la posición N del arreglo
- Mostrar un mensaje de solicitud de dato
- Almacenar el dato en la posición actual del arreglo

**En la ilustración siguiente se muestra la estructura general para realizar la lectura de un vector:**

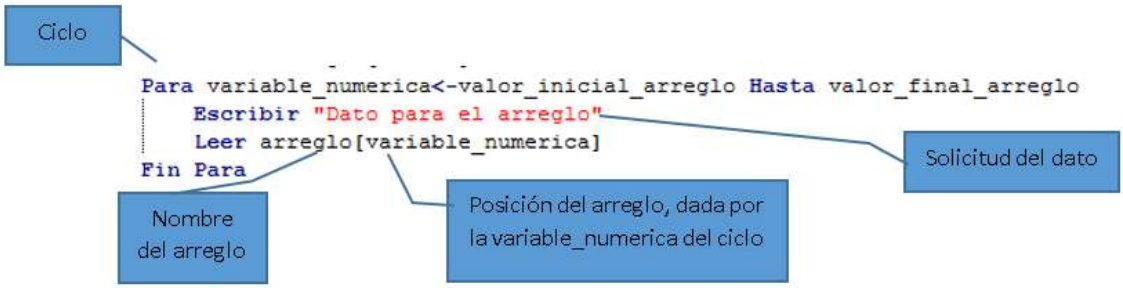



*Ilustración 56: Ejemplo lectura/llenado Vector*

**Ejemplo 1:** Se requiere almacenar 100 números enteros en un vector que se llame arreglo

La ilustración que se muestra a continuación, da solución al problema planteado

```
1  Algoritmo Arreglo100Datos
2      Entero arreglo[100]
3      Para i<-1 Hasta 100
4          Escribir "Dato para la posicion " i "del arreglo"
5          Leer arreglo[i]
6      Fin Para
7  FinAlgoritmo
```

*Ilustración 57: Ejemplo 1 plan Vector*

En la ilustración 58 se muestra la solución en el software **PSeInt** al ejemplo 1 del plan de programación Vector.

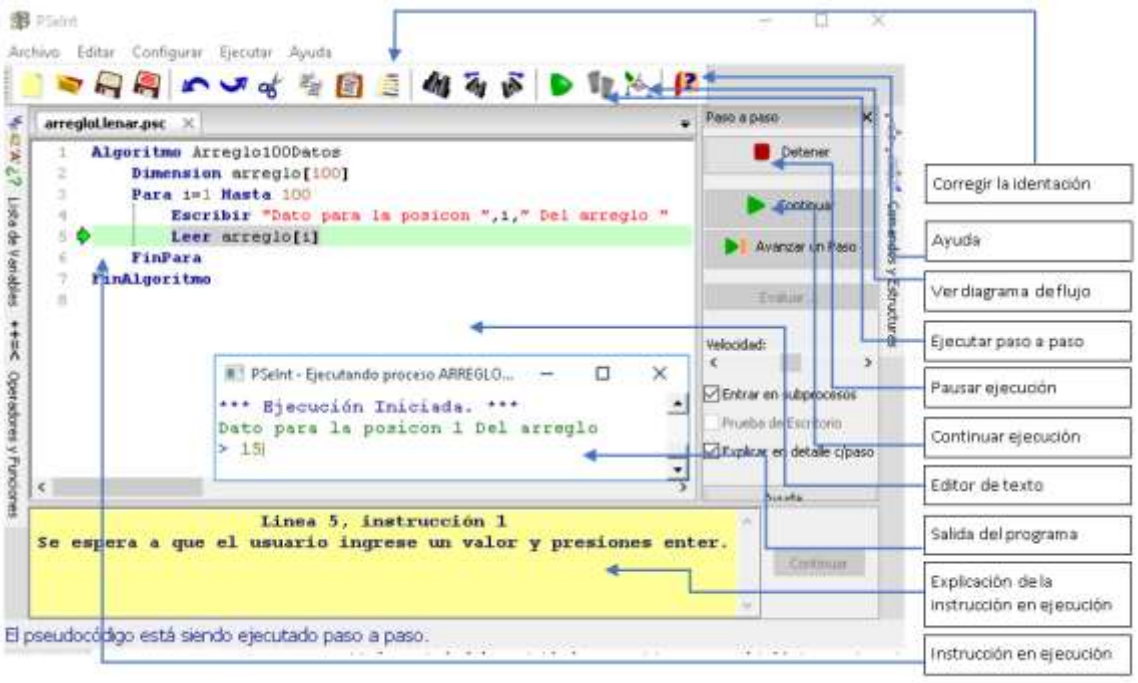



*Ilustración 58: Solución PSeInt ejemplo 1 Vector*

Las operaciones restantes del uso de los vectores se dejan para un estudio posterior.

# 5 METODOLOGÍA

El desarrollo de la metodología consistió de varios pasos, como primer paso se identificaron las estructuras de mayor uso en los primeros cursos de programación, las estructuras identificadas se enumeran a continuación; los planes de programación de cada una de ellas las pude encontrar en el capítulo anterior.

**Estructuras selectivas**

- SI – ENTONCES (condición simple)
- SI – ENTONCES / SINO (condición selectiva doble)
- SI MULTIPLE (Estructura selectiva usada para múltiples casos de selección)

## Estructuras de Repetición

- Plan ciclo MIENTRAS
- Plan ciclo REPETIR
- Plan ciclo PARA

## Contadores y acumuladores

Se agrega también en la tesis un plan Plan Arreglos/Vectores que no se probó en los estudiantes

Estas estructuras sirvieron como base para un segundo paso donde siguiendo la teoría de Van Merriënboer *et al.*(1994) para la construcción de planes de programación, se diseñaron para cada una de las estructuras identificadas sus correspondientes planes en base a las estructuras básicas previamente identificadas, cada una de las estructuras identificadas consta de distintas partes como bien se detalla en la sección 4 de esta tesis.

Estos planes de programación están diseñados pensando en que los estudiantes de primeros semestres de las carreras de las ciencias computacionales los vean fáciles y clarifique en ellos la forma de usar de manera adecuada cada una de las sentencias básicas de la programación, una vez diseñado el plan para una sentencia específica.

Además de los planes de programación se diseñaron un conjunto de ejercicios por completar que les sirvieran como base para aprender las sentencias que se revisaron previamente con los planes (ilustración 59), estos ejercicios se aplicaron como una tarea con el objetivo de revisar si se había comprendido bien las sentencias usadas para programar, se anexa el trabajo diseñado y aplicado a los estudiantes (ANEXO1).

```
1   Algoritmo SI_ENTONCES_SINO
2       Leer NUMERO
3       SI _____________Entonces
4       |   Imprimir "Numero positivo "
5       |
6       |   SI ____________Entonces
7       |   |   Imprimir "Numero negativo"
8       |   Sino
9       |   |   Imprimir "Nulo"
10      |   FinSi
11      FinSi
12  FinAlgoritmo
```

*Ilustración 59: Ejemplo ejercicio por completar*

Los ejercicios por completar se realizaron en la herramienta PseInt (Pablo Novara, Copyleft 2003-2017). Este software fue diseñado para ayudar a los estudiantes de cursos iniciales de programación a entender cómo es que funciona la misma de una manera más sencilla y práctica, esta herramienta permite crear algoritmos y diagramas de flujo que posteriormente se pueden ejecutar y mostrar los resultados obtenidos. Los algoritmos se pueden escribir en seudocódigo, haciendo uso de este programa de una forma cotidiana y como un primer contacto con la programación como forma de solución de problemas computacionales, esta herramienta es muy flexible a la hora de escribir seudocódigos, lo que les permite a los estudiantes no preocuparse tanto al inicio de la sintaxis particular de un lenguaje y centrarse más en la solución de un problema específico (Ilustración 60).

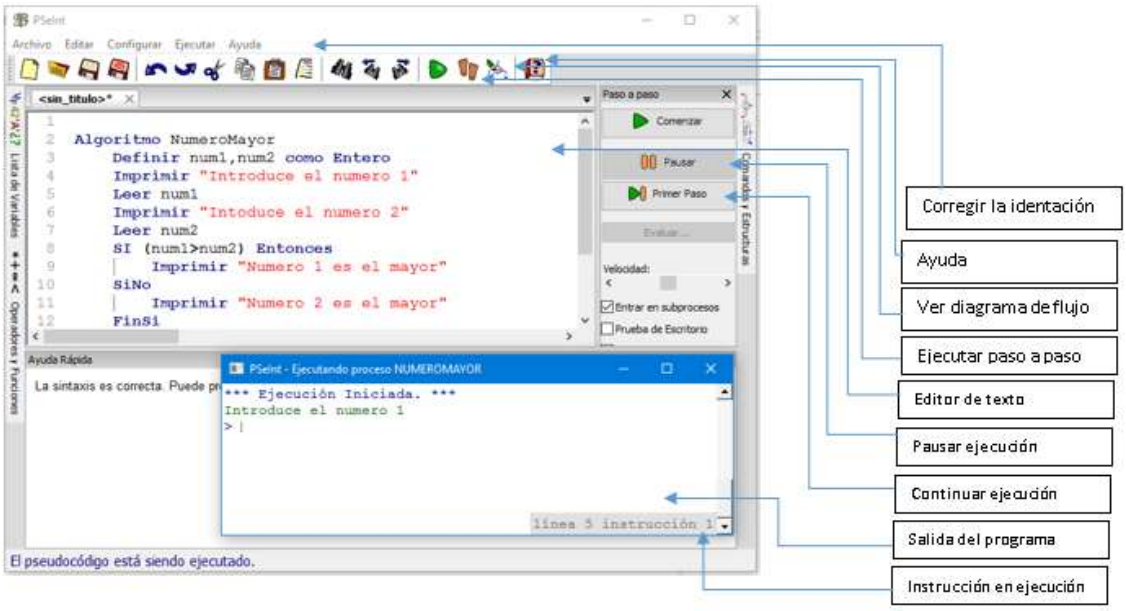



*Ilustración 60: Ejemplo solución PSeInt*

De igual manera, el software les permite a los estudiantes ver los errores que tiene un algoritmo, corregirlos y aprender de ellos antes de llegar a la ejecución de un problema libre de errores.

Junto con lo anterior, le provee a los estudiantes de diversas herramientas para ver cómo es que la computadora ejecuta un algoritmo paso por paso hasta llegar a un resultado final, esto proceso de revisión de cómo es que se van moviendo los datos dentro del algoritmo ejecutado, les permite identificar a los estudiantes que está pasando en la computadora con el algoritmo que se ejecuta y así comprender de una forma un poco más sencilla el proceso de ejecución de un programa.

El uso de todas las herramientas antes mencionadas se aplicó a un grupo experimental de 42 estudiantes de primer semestre de la carrera de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes.

Los 42 estudiantes del grupo experimental provienen de distintas instituciones de bachillerato del estado, cada uno de ellos con conocimientos sobre la programación

distintos, hay quienes provienen de instituciones donde se les forma en el área de programación y otros con conocimiento nulo en la materia, el 86% de los estudiantes provienen de bachilleratos con especialización en programación, en la siguiente tabla se puede ver un conteo de los alumnos que provienen de bachillerato con especialidad en programación, el documento completo se pude consultar en el ANEXO 5.

| # Alumnos | CBTIS 168, Especialidad en programación |
|---|---|
| 10/56 | Grupo de control |
| 7/50 | Grupo experimental |

*Ilustración 61: Porcentaje de alumnos con bases de programación*

Para el desarrollo y aplicación de la metodología se usaron los 4 pasos previamente descritos (ilustración 62), primeramente, se explicaron los planes de programación para que los estudiantes comprendieran cómo se comportan las estructuras dentro de la programación, posteriormente se mostraron ejercicios resueltos donde se incluía los planes previamente explicados.

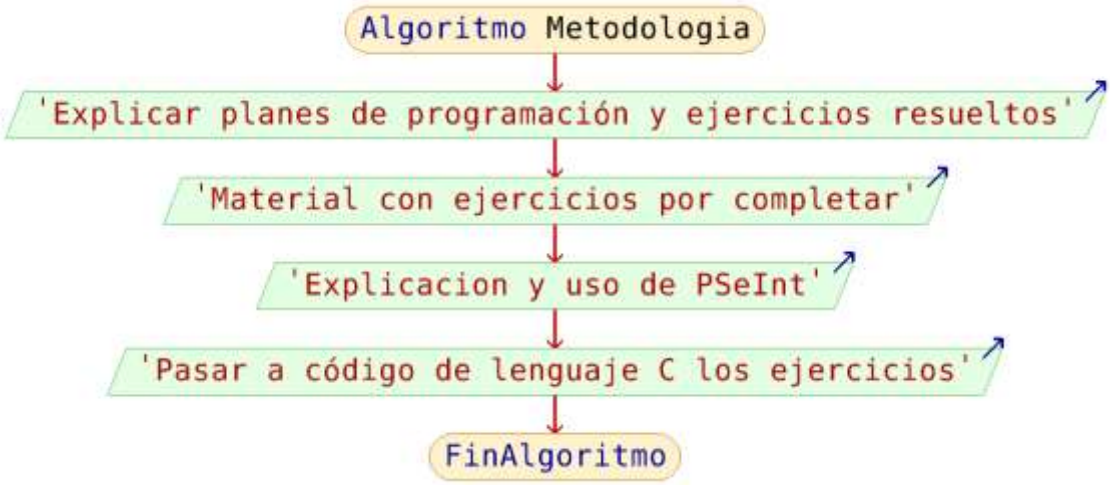



*Ilustración 62: Pasos de la metodología aplicada*

Posteriormente se diseñó **el material con los ejercicios por completar** donde los estudiantes rellenan espacios específicos con las construcciones de lenguaje que se

esperan aprendan en cada uno de los ejercicios que ahí se diseñaron, el objetivo de esas plantillas como ya se ha señalado anteriormente y se ha sustentado en la teoría base **de ejercicios por completar**, les ayuda a los estudiantes a centrarse en instrucciones específicas que se espera aprendan como producto de este tipo de ejercicios.

El material didáctico que aparece en el ANEXO 1, fue reproducido para entregar un juego físico a cada estudiante del grupo de estudio, de los 42 estudiantes pertenecientes al grupo experimental, 39 de los cuales regresaron la actividad contestada, se muestra una evidencia de ella en el ANEXO 2.

Una vez realizada esta actividad, se les solicito que como **un tercer paso** ahora usaran la herramienta de aprendizaje de PSeInt, en la cual capturaron cada uno de los ejercicios, esto con los objetivos siguientes:

a) Aprender como que se ejecutan los algoritmos en las computadoras, es decir, reforzar modelos mentales correctos entendiendo que es lo que pasa dentro de la memoria de la máquina, viendo cómo se mueven y se almacenan las variables, como trabajan las condiciones y los ciclos con en base a los algoritmos que ellos capturaron.

b) Constatar que sus respuestas estaban correctas y de no ser así que corrigieran las mismas para obtener el aprendizaje esperado.

Mediante el uso de esta metodología, se agrega evidencia de los trabajos entregados vía Moodle que es la plataforma educativa usada por el Centro de Ciencias Básicas de la Universidad Autónoma de Aguascalientes (ANEXO 3).

El instrumento usado para este fin constó con 6 ejercicios, cada uno de ellos con distinto grado de complejidad y cada uno más complejo y con más construcciones de lenguaje que el anterior, cada ejercicio consto de distintos apartados los cuales deberían de cumplir para cada caso, como se muestra en la Ilustración 62.

```
Algoritmo Hospital
    TipoEnf Es Entero;
    Edad Es Entero;
    Dias Es Entero;
    Costo Es Real;
    Escribir "Tipo de enfermedad 1/2/3/4 ";
    Leer________;
    Escribir "Edad del paciente ";
    Leer Edad;
    Escribir "Dias de hospital ";
    Leer Dias;
    Segun_________Hacer
        caso__:
            Costo=Dias*35;
        caso 2:
            Costo=Dias*16;
        caso__:
            Costo=Dias *20;
        caso__:
            _____________;
    FinSegun
    Si (Edad > 14  y___________)
        Costo=Costo*____;
    FinSi
    Escribir "El costo total del paciente es: ",_____;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

*Ilustración 63: Ejemplo material aplicado en la metodología*

**Como ultimo y cuarto paso de la metodología** se les pidió que, ahora ya corregidos y entendidos los ejercicios solicitados, escribieran código en lenguaje C para verificar que el funcionamiento de los ejercicios en un lenguaje de programación formal es igual al ya obtenido mediante las plantillas que completaron y mediante la herramienta de apoyo de PSeInt, este trabajo también se solicitó a los estudiantes que lo entregaran vía plataforma Moodle de la cual se agrega evidencia (ANEXO 4).

La metodología se aplicó durante el periodo de tiempo del segundo examen parcial (ver Ilustración 64), la aplicación de todas las fases de la metodología y en la recopilación de cada uno de las evidencias de los trabajos solicitados se llevó aproximadamente cuatro semanas. Teniendo alrededor de un 93% de las tareas cumplidas por los estudiantes del grupo experimental, cabe hacer mención que las tareas entregadas en el curso de lógica de programación llevan un porcentaje de calificación de la correspondiente a cada evaluación parcial.

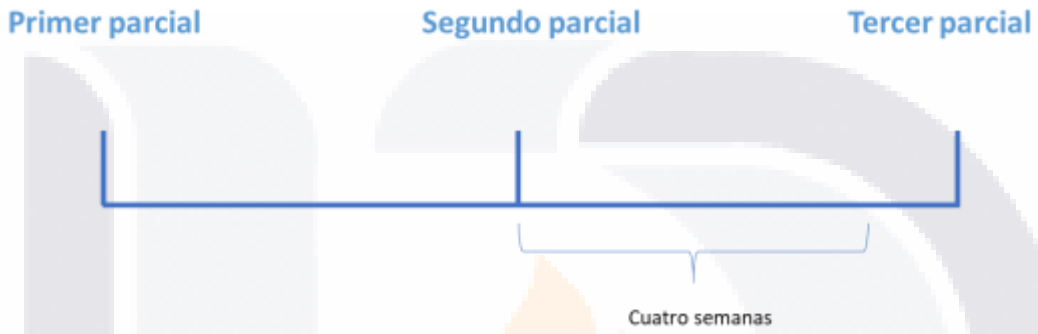



*Ilustración 64: Periodo de tiempo de aplicación de material*

Los temas que se incluyeron en la metodología de trabajo fueron los siguientes: de las estructuras de control selectivas, se diseñaron las tres que se trabajan ordinariamente en la programación, SI–ENTONCES (condición simple), SI–ENTONCES / SINO (condición selectiva doble), SI MULTIPLE (Estructura selectiva usada para múltiples casos de selección). De las estructuras de repetición se diseñaron y trabajaron las tres más comunes, ciclo MIENTRAS, ciclo REPETIR y ciclo PARA, dentro del uso de los ciclos es muy común usar Contadores y Acumuladores, por lo cual también de estos se diseñó un plan y se incluyó en los ejercicios.

### 5.1 Diseño de la investigación

Se utilizó un diseño pre-prueba, post-prueba con grupo de control, de acuerdo al diseño de la Ilustración 66

| Diseño experimental | | | | |
|---|---|---|---|---|
| $G_1$ | $O_1$ (primer parcial) | $O_2$ (2do. Parcial) | -- | $O_3$ (3er. Parcial) |
| $G_2$ | $O_4$ (primer parcial) | $O_5$ (2do. Parcial) | $X_1$ (Tratamiento Experimental) | $O_6$ (3er. Parcial) |

*Ilustración 65: Diseño experimental*

Dado que el tratamiento se aplicó posteriormente al segundo examen parcial, se cuenta con dos observaciones previas al mismo. La observación post-prueba corresponde al tercer examen parcial. Como ya se ha mencionado, ambos grupos son de estudiantes de nuevo ingreso de la Universidad Autónoma de Aguascalientes, ambos pertenecen a las carreras de Ingeniería en Sistemas Computacionales, de tal manera que ambos grupos vienen en condiciones semejantes (se anexan datos de bachillerato de origen, número de alumnos en el ANEXO 5)

Con el objetivo de controlar el estilo de enseñanza de los grupos, la docente que impartió y aplico la metodología al grupo experimental fue la misma que impartió clase al grupo de control, el contenido del programa de estudio, así como la complejidad de los exámenes aplicados en cada uno de los grupos fueron los mismos.

El propósito de esta tesis ha sido diseñar, desarrollar, aplicar y evaluar, si a través del uso de planes de programación, que son aplicados en el proceso de enseñanza aprendizaje de 1 de los 3 grupos de Ingeniería en Sistemas Computacionales que ingresan a la Universidad Autónoma de Aguascalientes en el periodo de ingreso Agosto 2017, se obtiene un grado mayor de aprovechamiento en la materia de lógica de programación. Este aprovechamiento, se pretende medir mediante una variable correspondiente al aprendizaje de la programación básica, misma que mide los resultados de los exámenes obtenidos posterior a la aplicación del experimento en el grupo experimental.

Este es un estudio longitudinal y cuantitativo, longitudinal debido a que pretende que se lleve a cabo durante el primer semestre de los estudiantes de nuevo ingreso de la carrera de ISC, cuantitativo ya que será un proceso secuencial y probatorio que pretende evaluar el nivel de aprovechamiento, así como el tiempo de resolución de problemas relacionados con la programación de los estudiantes de Ingeniería en Sistemas Computacionales de primer semestre.

La selección de los grupos se realizó en base a los docentes que se asignaron a estas materias durante el periodo ya mencionado, se prefirió que el docente fuera el mismo tanto para el grupo donde se aplicaron los planes de programación, así como el grupo de control, eso con el fin de evitar algún sesgo o desviación en los resultados obtenidos como resultado de la metodología de enseñanza usada por el facilitador/acompañante del conocimiento.

Se busca aplicar el experimento en 1 de los 3 grupos de nuevo ingreso. De los dos grupos restantes uno servirá como grupo de control para realizar un análisis comparativo de los resultados obtenidos como producto del uso de los planes de programación.

Es importante para el estudio que sean estudiantes de nuevo ingreso de la carrera de Ingeniería en Sistemas Computacionales, debido a que estos han pasado por un proceso previo de selección y de alguna manera se puede asumir que tienen características y aptitudes similares, así mismo a los estudiantes se les imparte la metería de lógica de programación (en la cual se pretende probar los planes de programación) dónde se busca que el estudiante desarrolle habilidades de programación.

El análisis de los resultados se hará mediante estadística descriptiva y pruebas de diferencias de medias entre muestras dependientes e independientes.

# 6 RESULTADOS

## 6.1 Estadística descriptiva

Posterior a la conducción del diseño experimental descrito en el capítulo anterior, consistente en la aplicación del tratamiento de plantillas basadas en los planes de programación identificados (a su vez, basados en el efecto de problemas por completar de la teoría de carga cognitiva), al final del semestre las calificaciones parciales, medidas con los instrumentos calibrados, fueron como se muestra a continuación (ver tablas 1, 2 & 3)

**Estadísticos**

| | | PrimerParcial Exp | PrimerParcial Ctrl |
|---|---|---|---|
| N | Válidos | 42 | 47 |
| | Perdidos | 5 | 0 |
| Media | | 78.1714 | 84.5426 |
| Mediana | | 81.5500 | 91.0000 |
| Moda | | 93.00 | 100.00 |
| Desv. típ. | | 13.26882 | 18.09901 |
| Varianza | | 176.062 | 327.574 |
| Mínimo | | 43.40 | 23.50 |
| Máximo | | 97.00 | 100.00 |

*Tabla 1: Estadísticos primer parcial grupo experimental*

**Estadísticos**

| | | SegundoParcialExp | SegundoParcialCtrl |
|---|---|---|---|
| N | Válidos | 42 | 47 |
| | Perdidos | 5 | 0 |
| Media | | 75.8095 | 67.6723 |
| Mediana | | 75.8750 | 75.0000 |
| Moda | | 90.00 | 93.50 |
| Desv. típ. | | 18.73394 | 22.72134 |
| Varianza | | 350.961 | 516.259 |
| Mínimo | | 21.60 | 10.50 |
| Máximo | | 99.25 | 96.00 |

*Tabla 2: Estadísticos segundo parcial grupo experimental*

**Estadísticos**

| | | GpoExp | GpoCtrl |
|---|---|---|---|
| N | Válidos | 42 | 47 |
| | Perdidos | 5 | 0 |
| Media | | 72.0167 | 52.6574 |
| Mediana | | 75.3750 | 51.9500 |
| Moda | | 100.00 | .00 |
| Desv. típ. | | 21.27686 | 33.60510 |
| Varianza | | 452.705 | 1129.303 |
| Asimetría | | -1.252 | -.279 |
| Error típ. de asimetría | | .365 | .347 |
| Curtosis | | 1.565 | -1.375 |
| Error típ. de curtosis | | .717 | .681 |
| Mínimo | | 10.50 | .00 |
| Máximo | | 100.00 | 99.25 |

*Tabla 3: Estadísticos tercer parcial grupo experimental*

Comparativamente, se observa que en ambos grupos la media fue decreciendo. En el grupo de control la media fue de 84.54, 67.67 y 52.65, respectivamente. Para el grupo experimental, las medias de cada parcial fueron 78.17, 75.80 y 72.01. El grupo de control inició con una media mayor (84.54), pero los resultados fueron decreciendo notoriamente hasta ser un valor reprobatorio en el tercer parcial. Por su parte, el grupo experimental inició con una media menor (78.17) pero el decremento fue marginalmente menor y la media del tercer parcial se mantuvo dentro del rango de notas aprobatorias.

Tablas 4 a 9. Histogramas de distribución de frecuencias, calificaciones grupos de control y experimental

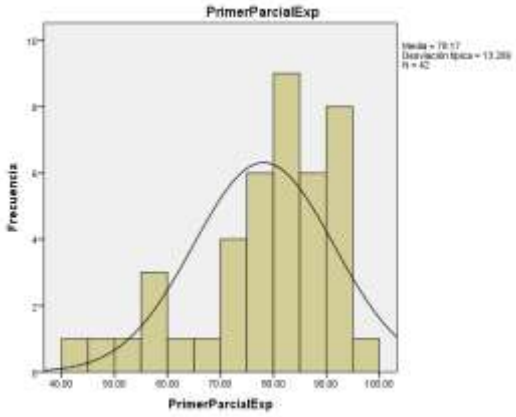



*Tabla 4: Histogramas primer parcial grupo experimental*

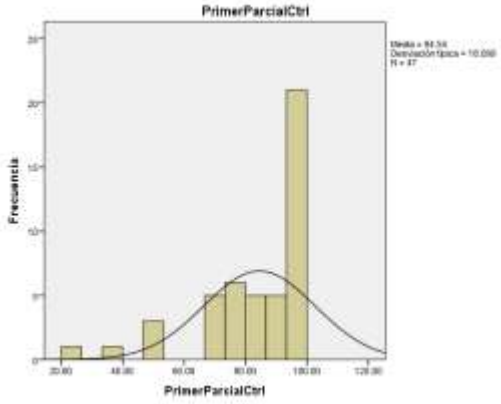



*Tabla 5: Histogramas primer parcial grupo control*

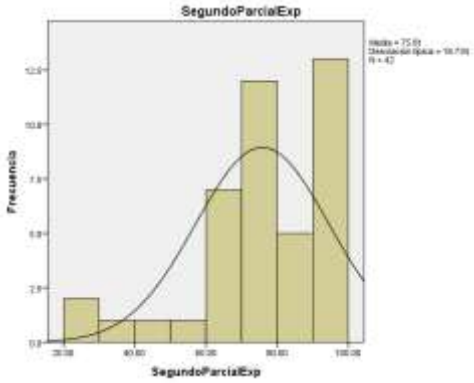



*Tabla 6: Histogramas segundo parcial grupo experimental*

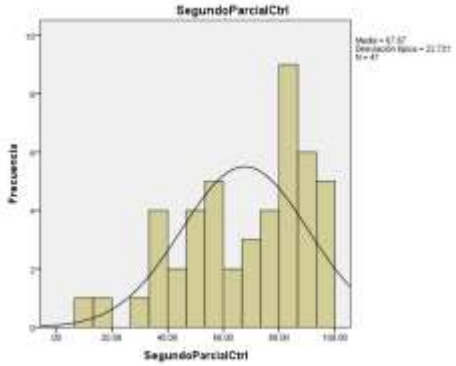



*Tabla 7: Histogramas segundo parcial grupo control*

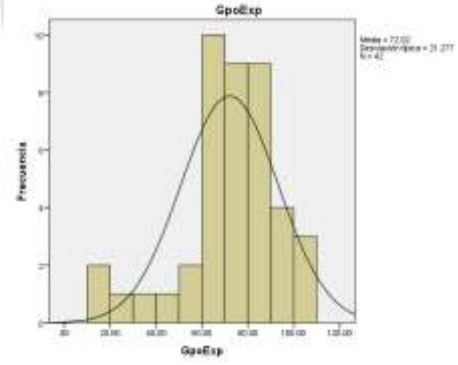



*Tabla 8: Histogramas tercer parcial grupo experimental*

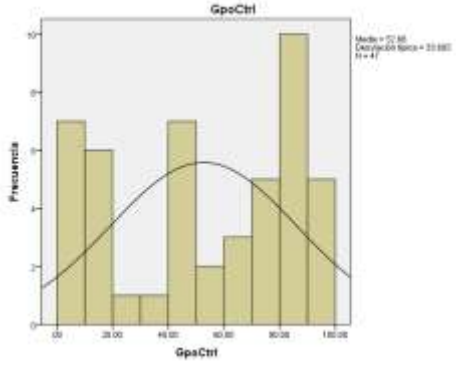



*Tabla 9: Histogramas tercer parcial grupo control*

Para el grupo de control, las distribuciones de frecuencias muestran (ver tabla 14 a 19 del anexo 11.6. Tablas de distribución de frecuencias de exámenes parciales, grupos experimental y de control) que un segmento notoriamente mayor de participantes obtuvo una alta calificación (19 casos están en el rango de 80 y 100), proporción que se mantuvo en el segundo parcial (20 casos). En el tercer parcial, las frecuencias en ese mismo rango disminuyeron (15 casos), pero no de manera significativa. Sin embargo, el número de participantes con calificación reprobatoria (es decir, menor a 65) en el tercer parcial (22 casos) se incrementó notoriamente mostrando una distribución con curtosis platicúrtica (ver tablas anexo 6).

Para el grupo experimental los casos de calificaciones en el rango entre 80 y 100 fueron 24 en el primer parcial, 18 en el segundo parcial y 16 en el tercero. Aunque se observa un decremento en todos los casos, éste puede considerarse bajo. En este grupo, los casos en el tercer parcial con calificaciones menores a 65 fueron solamente 8, lo cual representa una diferencia notable respecto a los 22 reprobados del grupo de control.

### 6.2 Comparación de medias

A continuación se muestran los resultados de comparación de medias entre los resultados de los tres exámenes parciales, tanto en la forma de comparación de muestras dependientes (es decir, los resultados de un mismo participante dentro del grupo, en cada parcial), así como de muestras independientes entre los grupos experimentales y de control, en los resultados del tercer examen parcial, que es en el que se espera que el efecto del tratamiento pedagógico por medio de las plantillas de planes de programación fuese medible.

Se aplicaron cuatro pruebas t, dos para cada grupo, para comparar tanto los resultados entre el primer examen parcial y el segundo, como los del segundo parcial contra el tercero. Se asumen distribuciones aproximadamente normales para los resultados de cada grupo (ver tablas).

Primer parcial vs. segundo parcial, muestras dependientes. Grupos experimentales y de control.

**Prueba de muestras relacionadas**

| | | Diferencias relacionadas | | | | | t | gl | Sig. (bilateral) |
|---|---|---|---|---|---|---|---|---|---|
| | | Media | Desviación típ. | Error típ. de la media | 95% Intervalo de confianza para la diferencia Inferior | Superior | | | |
| Par 1 | PrimerParcialExp - SegundoParcialExp | 2.36190 | 10.41775 | 1.60749 | -.88450 | 5.60831 | 1.469 | 41 | .149 |
| Par 2 | PrimerParcialCtrl - SegundoParcialCtrl | 16.87021 | 11.64067 | 1.69797 | 13.45238 | 20.28804 | 9.936 | 46 | .000 |

*Tabla 10: Muestras relacionadas primer parcial*

Segundo parcial vs. tercer parcial, muestras dependientes. Grupos experimentales y de control.

**Prueba de muestras relacionadas**

| | | Diferencias relacionadas | | | | | t | gl | Sig. (bilateral) |
|---|---|---|---|---|---|---|---|---|---|
| | | Media | Desviación típ. | Error típ. de la media | 95% Intervalo de confianza para la diferencia Inferior | Superior | | | |
| Par 1 | SegundoParcialExp - FinalExp | 3.79286 | 11.93130 | 1.84104 | .07480 | 7.51091 | 2.060 | 41 | .046 |
| Par 2 | SegundoParcialCtrl - FinalCtrl | 15.01489 | 17.59314 | 2.56622 | 9.84936 | 20.18043 | 5.851 | 46 | .000 |

*Tabla 11: Muestras relacionadas segundo parcial*

Tabla resumen comparación de medias, muestras relacionadas (dependientes)

| Muestras dependientes | | |
|---|---|---|
| Medición | **1er parcial vs. 2do parcial** | **2do parcial vs 3er parcial** |
| Grupo de Control | p=0.000 | p=0.000 |
| Grupo Experimental | p=0.149 | p=0.046 |

*Tabla 12: Muestras relacionadas dependientes*

Las pruebas de comparación de medias con muestras dependientes indican que en el grupo de control hubo diferencia significativa en el comportamiento de los resultados de las tres calificaciones parciales. En otras palabras, en cada distribución, las calificaciones se comportaron de manera diferente. Para el grupo experimental, las comparaciones entre las calificaciones parciales intragrupo muestran que no hubo diferencia significativa, salvo una diferencia estadística limítrofe (p=0.046) en la comparación del segundo y tercer parcial. Esto sugiere que el comportamiento de las calificaciones de este grupo fue homogéneo.

Tercer parcial grupo de control vs. grupo experimental, muestras independientes.

**Prueba de muestras independientes**

| | | Prueba de Levene para la igualdad de varianzas | | Prueba T para la igualdad de medias | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | 95% Intervalo de confianza para la diferencia | |
| | | F | Sig. | t | gl | Sig. (bilateral) | Diferencia de medias | Error típ. de la diferencia | Inferior | Superior |
| Calific | Se han asumido varianzas iguales | 18.237 | .000 | 3.203 | 87 | .002 | 19.35922 | 6.04482 | 7.34448 | 31.37396 |
| | No se han asumido varianzas iguales | | | 3.281 | 78.748 | .002 | 19.35922 | 5.89970 | 7.61559 | 31.10285 |

*Tabla 13: Tercer parcial, muestras independientes*

La prueba de comparación de medias de muestras independientes entre los resultados del tercer parcial (es decir, posterior al tratamiento) de los grupos de control y experimental arroja un valor p=0.02, lo cual indica una diferencia estadísticamente significativa entre ambos grupos.

Visualmente, esto puede corroborarse comparando los histogramas de frecuencias de los resultados del tercer parcial de ambos grupos (ver tablas 4 a 9) en donde se observa que la curtosis de la distribución de los resultados del tercer parcial del grupo experimental es *leptocúrtica* y la del grupo de control es *platicúrtica*. Esto, agregado a la comparación directa de los valores de las medias de ambos grupos (Control = 52.65; Experimental = 72.01) sugiere la existencia de un efecto positivo derivado del uso del tratamiento pedagógico mediante el uso de plantillas basadas en planes de programación.

# 7 CONCLUSIONES

## 7.1 Conclusiones sobre los resultados del estudio

Derivado de las pruebas aplicadas tanto al grupo experimental como al grupo de control, se puede observar que las pruebas de comparación de medias dependientes (es decir, las pruebas pareadas) del primer y segundo parcial no mostraron diferencias. En otras palabras, no hubo cambios significativos (p=0.149) en el rendimiento académico entre estas dos observaciones. Por otro lado, la prueba pareada de comparación de medias entre el segundo y tercer parcial de éste mismo grupo experimental *sí* muestra un cambio estadístico significativo (p=0.046), lo cual proporciona indicios de un efecto debido al tratamiento basado en planes de programación y problemas por completar, ya que éste tuvo lugar entre estas dos observaciones. Este efecto positivo también es visible mediante la prueba de comparación de medias del tercer parcial entre los grupos experimental y de control (p=0.002). Como confirmación adicional, las medias de ambos grupos tuvieron una diferencia de casi un 20% (19.3593 puntos).

Derivado del análisis anterior, puede asumirse que el uso de ejercicios basados en los planes de programación identificados, y éstos a su vez basados en problemas por completar sugeridos por la teoría de carga cognitiva, tuvieron un efecto positivo en el aprendizaje de los alumnos. El efecto predicho por la Teoría, en el sentido de que la carga cognitiva extrínseca sobre la memoria de corto plazo al utilizar problemas por completar es menor al uso de estrategias de solución de tipo “prueba y error”, se corrobora mediante los resultados obtenidos. Por otro lado, debe mencionarse que no es posible aún hacer una generalización de los resultados, dado que el diseño experimental no contó con aleatorización, pero consideramos que éstos son fiables, debido a la homogeneidad de los grupos conformados en el estudio (es decir, el proceso de admisión de la UAA ya maneja un proceso de aleatorización para la conformación de los grupos, además existe la consideración de que la experiencia previa en programación es similar, e incluso menor en el grupo experimental).

Creemos que futuras réplicas de este estudio, mediante la depuración y extensión del material didáctico utilizado, pueden confirmar los resultados obtenidos, lo cual a su vez será directamente benéfico para los estudiantes universitarios de programación básica.

### 7.2 Conclusiones acerca de los objetivos

El objetivo general de la tesis fue "*diseñar, desarrollar y probar material didáctico basado en identificación de planes de programación con el propósito de medir su efectividad en el aprendizaje de la programación en alumnos de primer año del programa educativo de Ingeniería en Sistemas Computacionales de la Universidad Autónoma de Aguascalientes*" se ha cumplido con resultados positivos.

Se llevo a cumplimiento cada uno de los objetivos propuestos al inicio de este trabajo, se identificaron los planes de programación básicos de uso común en la solución de problemas de programación básica, detallados en el capítulo 4 de esta tesis.

En el mismo capítulo 4, se muestra el material didáctico basado en los planes de programación previamente identificados y con los cuales se realizó la prueba del material en el grupo experimental.

Así pues, dando respuesta a las preguntas de investigación planteadas en el presente trabajo se puede concluir que si es posible identificar y documentar un conjunto de planes de programación que nos sirva como material didáctico para estudiantes de nuevo ingreso que comienzan con la programación.

De igual forma, se pude concluir que el diseño de material didáctico realizado con el fin de proporcionar a los estudiantes un mayor número de habilidades para la solución de problemas de programación, dio un efecto positivo en los estudiantes.

Se puede señalar que, sí es posible aplicar estrategias didácticas donde basándonos en la teoría de la carga cognitiva, y que ésta nos apoya en la elaboración de los materiales didácticos los cuales facilitan la adquisición de habilidades para la resolución de problemas de programación en los estudiantes de nuevo ingreso a esta área del conocimiento.

- **Plan de programación**: se define como una secuencia de instrucciones para solucionar problemas que se desarrollan de la misma manera siempre, es decir, las soluciones siguen un plan uniforme.
- **Carga cognitiva intrínseca**: se relaciona con la dificultad inherente de aquello que se está tratando de aprender (Sweller 1994, 2010; Sweller & Chandler 1994). En términos simples, la carga intrínseca puede ser descrita como el tipo de carga cognitiva "necesaria".
- **Carga cognitiva extrínseca**: "*La carga cognitiva extrínseca se relaciona con cómo el contenido se enseña. Según Van Merrienboer y Sweller, "La carga cognitiva extrínseca es una carga que no necesariamente produce aprendizaje y que puede ser alterada por intervenciones basadas en la instrucción" (2005, p. 105). En pocas palabras, la carga extrínseca es la carga "mala" porque no contribuye directamente al aprendizaje. Los teóricos consideran que un diseño de instrucción será más efectivo cuando se encarga de minimizar la carga extrínseca con el objetivo de liberar la capacidad de memoria de trabajo*".
- **Carga cognitiva pertinente**: "*La carga cognitiva pertinente se refiere a la carga impuesta a la memoria de trabajo por el propio proceso de aprendizaje, es decir, sería el proceso de transferir información a la memoria a largo plazo a través de la construcción de esquemas (Sweller, van Merrienboer & Paas 1998, p. 259 . Por esta razón, la carga cognitiva pertinente puede ser entendida como un tipo de carga "buena"*".
- **Memoria de corto plazo**: mecanismo de memoria que nos permite retener una cantidad limitada de información durante un periodo corto de tiempo. La memoria a corto plazo retiene temporalmente la información procesada, tanto si luego se desvanece, como si después pasa a la memoria a largo plazo. Así, la memoria a corto plazo tiene dos propiedades principales: una capacidad limitada y una duración finita.
- **Memoria de largo plazo**: también llamada memoria inactiva o memoria secundaria, es un tipo de memoria que almacena recuerdos por un plazo de tiempo mayor a seis meses, sin que se le presuponga límite alguno de capacidad o duración (León-Carrión, 1995, p 333-335).

- **Memoria de trabajo**: es el sistema de memoria donde “pequeñas” cantidades de información se almacenan durante un muy breve periodo de tiempo
- **Esquemas (schema):** organiza elementos de información de acuerdo a cómo serán usados. Según la teoría de esquemas, para llegar a dominar una habilidad cognitiva hay que ir construyendo esquemas cada vez más complejos a través de insertar elementos de esquemas inferiores en esquemas superiores.

# 8 REFERENCIAS

A Tool to Support the Use of Part-Complete Solutions in the Learning of Programming - GarnerEBKATool.pdf. (s/f). Recuperado a partir de http://www.proceedings.informingscience.org/IS2001Proceedings/pdf/GarnerEBKATool.pdf

Ala-Mutka, K. (2004). Problems in learning and teaching programming – a literature study for developing visualizations in the Codewitz-Minerva project, Codewitz needs analysis, http://www.cs.tut.fi/~edge/literature_study.pdf.

Ananya, H. A., Hegde, I. A., Joshi, A. G., & Kumar, V. (2016). Ranking Student Ability and Problem Difficulty Using Learning Velocities. En S. Berretti, S. M. Thampi, & P. R. Srivastava (Eds.), Intelligent Systems Technologies and Applications (pp. 191–199). Springer International Publishing. https://doi.org/10.1007/978-3-319-23036-8_17

Bashir, G. M. M., & Hoque, A. S. M. L. (2016). An effective learning and teaching model for programming languages. Journal of Computers in Education, 1–25. https://doi.org/10.1007/s40692-016-0073-2

Bennedsen, J., & Caspersen, M. E. (2008). Exposing the Programming Process. En J. Bennedsen, M. E. Caspersen, & M. Kölling (Eds.), Reflections on the Teaching of Programming (pp. 6–16). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77934-6_2

Bornat, R., Dehnadi, S., & Simon. (2008). Mental models, Consistency and Programming Aptitude. Presentado en Psychology of Programming interested Group (PPIG).

Bruce, C. (2004). Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. *Journal of Information Technology Education*, *3*, 144.

Centre for Education Statistics and Evaluation. (Agosto 2017). Cognitive load theory: Research that teachers really need to understand. © NSW Department of Education, 1, 1-12.

Chang, K.-E., Chiao, B.-C., & Hsiao, R.-S. (1996a). A programming learning system for beginners — A completion strategy approach. En *Intelligent Tutoring Systems*

(pp. 623–631). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61327-7_162

Ciernaiak, G., Scheiter, K., & Gerjets, P. (2009). Explaining the split-attention effect: Is the reduction of extraneous cognitive load accompanied by an increase in germane cognitive load? Computers in Human Behavior

Cooper, G & Sweller, J (1987), 'Effects of schema acquisition and rule automation on mathematical problem-solving transfer', *Journal of Educational Psychology*, vol. 79, no. 4, pp. 347-362.

Crissman, J 2006, *The design and utilization of effective worked examples: A meta-analysis,* unpublished doctoral thesis, University of Nebraska, Lincoln N.E.

Dr. Osvaldo Cairo Battistutti. (2005). *Metodología de la programación* (3a ed.). México,: Alfaomega.

Fleming, N., & Baume, D. (2006). Learning styles again: VARKing up the right tree! Educational Developments, SEDA Ltd, Issue 7.4, (pp. 4–7).

Gallego-Durán, F. J., Molina-Carmona, R., & Llorens-Largo, F. (2016). An Approach to Measuring the Difficulty of Learning Activities. En P. Zaphiris & A. Ioannou (Eds.), *Learning and Collaboration Technologies* (pp. 417–428). Springer International Publishing. https://doi.org/10.1007/978-3-319-39483-1_38

Garner, S. K. (2001). A Tool to Support the Use of Part-Complete Solutions in the Learning of Programming. Recuperado a partir de http://ro.ecu.edu.au/ecuworks/4578

Guzdial, M., & Soloway, E. (2002). Log on education: teaching the Nintendo generation to program. *Communications of the ACM, 45*(4), 17–21.

Hashim, N., & Salam, S. (2009). Integration of Visualization Techniques and Completion Strategy to Improve Learning in Computer Programming. En *2009 International Conference of Soft Computing and Pattern Recognition* (pp. 665–669). https://doi.org/10.1109/SoCPaR.2009.131

Iqbal, S., & Harsh, O. K. (2013). A self-review and external review model for teaching and assessing novice programmers. *International Journal of Information and Education Technology, 2*(3), 120–123.

Jenkins, T. (2002). On the difficulty of learning to program. Loughborough University: LTSN Centre of information and computer sciences.

John Sweller, Paul Ayres y Salva Kalyuga. (2011). Cognitive Load Theory. New York, NY 10013, USA: Springer.

Kelleher, C. P., & Pausch, R. (2005). Lowering the Barriers to Programming: a survey of programming environments and languages for novice programmers. *ACM Computing surveys (CSUR)*, *37*(2), 83–137.

Kölling, M., & Barnes, D. J. (2008). Apprentice-Based Learning Via Integrated Lectures and Assignments. En J. Bennedsen, M. E. Caspersen, & M. Kölling (Eds.), *Reflections on the Teaching of Programming* (pp. 17–29). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77934-6_3

Kujansuu E. (2006). Using program visualisation learning objects with non-major students with different study background. Presentado en In Proceedings of Methods, Materials and Tools for Programming Education conference, Tampere, Finland.

León-Carrión, J. (1995). *Manual de neuropsicología humana*. Siglo XXI de España Editores, S.A. Recuperado a partir de https://books.google.es/books?id=Jq4vXlrZrLkC

Malik, S. I., & Coldwell-Neilson, J. (2016). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 1–32. https://doi.org/10.1007/s10639-016-9474-0

Meisalo, V., Suhonen, J., Sutinen, E., & Torvinen, S. (2002). *Formative evaluation scheme for a web-based course design* (pp. 130–134). Proceedings of the 7thITiCSE, ACM, University of Aarhus, Denmark.

Merrienboer, J. J. G. V., & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science*, *16*(3), 251–285. https://doi.org/10.1007/BF00120253

Merriënboer, J. J. G. van, Krammer, H. P. M., & Maaswinkel, R. M. (1994a). Automating the Planning and Construction of Programming Assignments for Teaching Introductory Computer Programming. En *Automating Instructional Design, Development, and Delivery* (pp. 61–77). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-78389-0_4

Merriënboer, J. J. G. van, Krammer, H. P. M., & Maaswinkel, R. M. (1994b). Automating the Planning and Construction of Programming Assignments for Teaching Introductory Computer Programming. En R. D. Tennyson (Ed.), *Automating Instructional Design, Development, and Delivery* (pp. 61–77). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-78389-0_4

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, *7*(1), 55–66.

Mohorovicic, S., & Strcic, V. (2011). *An overview of computer programming teaching methods* (pp. 47–52). Proceedings of Central European Conference on Information and Intelligent Systems, CECIIS, Croatia.

Mow, I. T. C. (2008). Issues and Difficulties in Teaching Novice Computer Programming. *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*, 199–204.

Ogar, O., Shabalina, O., Davtyan, A., & Kizim, A. (s/f). Mastering Programming Skills with the Use of Adaptive Learning Games. En *SpringerLink* (pp. 144–155). Springer International Publishing. https://doi.org/10.1007/978-3-319-11854-3_14

Oliver, R. (1993). Measuring Hierarchical Levels of Programming Knowledge. *Journal of Educational Computing Research*, *9*(3), 299–312. https://doi.org/10.2190/0LGX-M45X-2WBK-B7A6

Papp-Varga, Z., Szlávi, P., & Zsakó, L. (2008). ICT teaching methods – Programming languages. Annales Mathematicae et Informaticae, 35(1), 163–172.

Parham, J., Gugerty, L., & Stevenson, D. E. (2010). Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving. En *Proceedings of the 41st ACM technical symposium on Computer science education*. Milwaukee, WI, USA: ACM New York, NY, USA.

Park, N., & Ko, Y. (2012). Computer Education's Teaching-Learning Methods Using Educational Programming Language Based on STEAM Education. En J. J. Park, A. Zomaya, S.-S. Yeo, & S. Sahni (Eds.), *Network and Parallel Computing* (pp. 320–327). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35606-3_38

Paas, F 1992, 'Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach', *Journal of Educational Psychology*, vol. 84, no. 4, pp. 429-434.

Paas, F & van Merrienboer, J 1994, 'Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive load approach', *Journal of Educational Psychology*, vol. 86, no. 1, pp. 122-133.

Pashler, H., McDaniel, M., & Bjork, R. (2008). Learning styles concepts and evidence. *Psychological Science in the Public Interest, 9*(3), 105–119.

Pillay, H 1994, 'Cognitive load and mental rotation: Structuring orthographic projection for learning and problem solving', *Instructional Science*, vol. 22, no. 2, pp. 91-113.

Quilici, J & Mayer, R 1996, 'Role of examples in how students learn to categorize statistics word problems', *Journal of Educational Psychology*, vol. 88, no. 1, pp. 144-161.

Schraw, G., & Moshman, D. (1995). Metacognitive theories. *Educational Psychology Review*, *7*(4), 351–371. https://doi.org/10.1007/BF02212307

Shuhidan, S.M. (2012). *Probing the minds of novice programmers through guided learning*, PhD thesis, retrieved July 2013, RMIT University: Australia.

Sweller, J., & Cooper, G. A. (1985). The Use of Worked Examples as a Substitute for Problem Solving in Learning Algebra. *Cognition and Instruction*, *2*(1), 59–89.

Sweller, J, van Merrienboer, J & Paas, F (1998), 'Cognitive architecture and instructional design', *Educational Psychology Review*, vol. 10, no. 3, pp. 251-296.

Sweller, J 2010, 'Element interactivity and intrinsic, extraneous and germane cognitive load', *Educational Psychology Review*, vol. 22, no. 2, pp. 123-138.

Sweller, J 2016, 'Story of a research program', *Education Review*, vol. 23, pp. 1-18.

Sweller, J & Chandler, P 1994, 'Why some material is difficult to learn', *Cognition and Instruction*, vol. 12, no. 3, pp. 185-233.

Tavares, J., Brzezinski, I., Huet, I., Cabreal, A., & Neri, D. (2001). *Having coffee with professors and students to talk about higher education pedagogy and academic success*. Proceedings of the 24th International HERDSA conference: Newcastle.

Trætteberg, H., Mavroudi, A., Giannakos, M., & Krogstie, J. (s/f). Adaptable Learning and Learning Analytics: A Case Study in a Programming Course. En *SpringerLink* (pp. 665–668). Springer International Publishing. https://doi.org/10.1007/978-3-319-45153-4_87

Tuovinen, J & Sweller, J 1999, 'A Comparison of cognitive load associated with discovery learning and worked examples', *Journal of Educational Psychology*, vol. 91, no. 2, pp. 334-341.

Van Merriënboer, J. J. G. (1990). Strategies for Programming Instruction in High School: Program Completion vs. Program Generation. *Journal of Educational Computing Research*, *6*(3), 265–285. https://doi.org/10.2190/4NK5-17L7-TWQV-1EHL

Van Merriënboer, J. J. G. (1990). Strategies for Programming Instruction in High School: Program Completion vs. Program Generation. *Journal of Educational Computing Research*, *6*(3), 265–285. https://doi.org/10.2190/4NK5-17L7-TWQV-1EHL

Van Merriënboer, J. J. G., & De Croock, M. B. M. (1992). Strategies for Computer-Based Programming Instruction: Program Completion vs. Program Generation. *Journal of Educational Computing Research*, *8*(3), 365–394. https://doi.org/10.2190/MJDX-9PP4-KFMT-09PM

Van Merrienboer, J. J. G., Krammer, H. P. M., & Maaswinkel, R. M. (1994). Automating the planning and construction of programming assignments for teaching introductory computer programming. In R. D. Tennyson (Ed.), Automating Instructional Design, Development, and Delivery (NATO ASI Series F, Vol. 119) (pp. 61-77): Springer Verlag, Berlin.

Wang, F. L., Fong, J., Choy, M., & Wong, T.-L. (2007). Blended Teaching and Learning of Computer Programming. En H. Leung, F. Li, R. Lau, & Q. Li (Eds.), *Advances in Web Based Learning – ICWL 2007* (pp. 606–617). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78139-4_53

Wiedenbeck, S., LaBelle, D., & Kain, V. (2004). Factors affecting course outcomes in introductory programming. Institute of Technology, Carlow, Ireland.

Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin, 28*(3), 17–22.

Yuxiang, S. (2012). The Application of Subjective Teaching Method in Computer Programming Courses Teaching. En L. Zhang & C. Zhang (Eds.), *Engineering Education and Management* (pp. 665–669). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24820-7_106

# 9 ANEXOS

**Anexo 1: Ejercicios de autocompletar aplicados.**

Los siguientes ejercicios fueron aplicados como tarea al grupo experimental.

Realiza los siguientes ejercicios como se va pidiendo en cada una de las descripciones

1. Completa cada uno de los ejercicios colocando en la línea lo sea necesario para dar solución al ejercicio planteado:

**Ejercicio 1:** Se necesita un algoritmo tal que, dados como datos la matrícula de un alumno, la carrera en la que está inscrito, su semestre y su promedio; determine si el mismo es apto para pertenecer a alguna de las facultades menores que tiene la universidad. Si el alumno es aceptado teniendo en cuenta las especificaciones que se listan abajo, se debe imprimir su matrícula, carrera y la palabra “aceptado”.

Especificaciones para pertenecer a las facultades menores:

| | |
|---|---|
| Economía: | Semestre > 6 y promedio > 8.8 |
| Computación: | Semestre > 6 y promedio > 8.5 |
| Administración: | Semestre > 5 y promedio > 8.5 |
| Contabilidad: | Semestre > 5 y promedio > 8.5 |

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Aceptado
    mat Es Entero;
    carr Es Entero;
    sem Es Entero
    prom Es Real;
    Escribir "Cual es la matricula del alumno ";
    Leer mat;
    Escribir "Cual es la carrera del alumno";
    Escribir "1=Economia/2=Computacion/3=Administracion/4=Contabilidad";
    Leer carr;
    Escribir "Cual es el semestre del alumno ";
    Leer sem;
    Escribir "Cual es el promedio del alumn ";
    Leer prom;
    Segun_____Hacer
        caso 1:
            si (_____y prom >=8.8)
                Escribir mat," ",sem," Aceptado";
            FinSi
        caso __:
            si (sem>6 y__________)
            _____________________________________
            FinSi
        caso __: caso __:
            si (_________________)
                Escribir mat," ",sem," Aceptado";
            FinSi
    FinSegun
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.

c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 2:** En un hospital se ha hecho un estudio sobre los pacientes registrados durante los últimos 10 años, con el objeto de hacer una aproximación de los costos de internación por paciente. Se obtuvo un costo promedio diario según el tipo de enfermedad que aqueja al paciente. Además se pudo determinar que en promedio todos los pacientes con edad entre 14 y 22 años implican un costo adicional del 10%. La siguiente tabla expresa los costos diarios, según el tipo de enfermedad.

| Tabla | |
|---|---|
| Tipo de Enfermedad | Costo: paciente/día |
| 1 | 35 |
| 2 | 16 |

| | |
|---|---|
| 3 | 20 |
| 4 | 32 |

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Hospital
    TipoEnf Es Entero;
    Edad Es Entero;
    Dias Es Entero;
    Costo Es Real;
    Escribir "Tipo de enfermedad 1/2/3/4 ";
    Leer________;
    Escribir "Edad del paciente ";
    Leer Edad;
    Escribir "Dias de hospital ";
    Leer Dias;
    Segun_________Hacer
        caso__:
            Costo=Dias*35;
        caso 2:
            Costo=Dias*16;
        caso__:
            Costo=Dias *20;
        caso__:
            _____________;
    FinSegun
    Si (Edad > 14  y___________)
        Costo=Costo*____;
    FinSi
    Escribir "El costo total del paciente es: ",_____;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.

c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 3:** Un vendedor ha hecho una serie de ventas y desea conocer aquellas de $200 o menos, las mayores a $200 pero inferiores a $400, y el número de ventas de $400 o superiores a tal cantidad. Haga un diagrama de flujo que le proporcione al vendedor esta información después de haber leído los datos de entrada.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Ventas
    N,i,venta,vmenor200,ventre200y400,vmas400 Es Entero;
    vmenor200=0;
    ventre200y400=0;
    vmas400=0;
    Escribir "Cuantas son el total de ventas a leer ";
    Leer N;
    para i desde___hasta___Hacer
        Escribir "Cual es el monto de la venta ";
        Leer venta;
        Si (_________)
            vmenor200=_________+1;
        SiNo
            Si (venta <400)
                ventre200y400=_____________
            SiNo
                ____________________
            FinSi
        FinSi
    FinPara
    Escribir "Ventas de menos de 200 son: ",_________;
    Escribir "Ventas entre 200 y 400 son: ",ventre200y400;
    Escribir ____________________________________________
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.

c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 4:** Supóngase que en una reciente elección hubo cuatro candidatos (con identificadores 1, 2, 3, 4). Usted habrá de encontrar, mediante un programa, el número de votos correspondiente a cada candidato y el porcentaje que obtuvo respecto al total de los votantes. El usuario tecleará los votos de manera desorganizada, tal y como se obtuvieron en la elección, el final de datos está representado por un cero.

Observe, como ejemplo, la siguiente lista:

1 3 1 4 2 2 1 4 1 1 1 2 1 3 1 4 0

Donde 1 representa un voto para el candidato 1; 3 un voto para el candidato 3; y así sucesivamente.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo votos
    C1,C2,C3,C4,totVotos,voto Es Entero;
    porm1,prom2,prom3,prom4 Es Real;
    C1=0;
    C2=0;
    _____
    _____
    Repetir
        Escribir "Voto para cadidato 1/2/3/4 0=Terminar";
        Leer voto;
        Segun_____hacer
            caso 1:
                C1=C1+1;
            caso 2:
                C2=C2+1;
            caso__:
                C3=C3+1;
            caso 4:
                _______;
        FinSegun
    Hasta Que_______:
    totVotos=__________;
    prom1=__/totVotos*100;
    prom2=C2/________*100;
    prom3=C3/totVotos*___;
    _____________________
    Escribir "Total de votos para candiato 1-> ",__," Porcentaje ",prom1,"%";
    Escribir "Total de votos para candiato 2-> ",C2," Porcentaje ",_____,"%";
    Escribir "Total de votos para candiato 3-> ",C3," Porcentaje ",prom3,"%";
    Escribir "Total de votos para candiato 4-> "                              ;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.

c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 5:** Se requiere saber lo que hay que pagar por un conjunto de llamadas telefónicas. Por cada llamada se ingresa el tipo (Internacional=1, Nacional=2, Local=3) y la duración en minutos. El criterio que se sigue para calcular el costo de cada llamada es el siguiente:

*Internacional:* 3 primeros minutos $7.59 Cada minuto adicional $3.03
*Nacional: 3* primeros minutos $1.20 Cada minuto adicional $0.48
*Local:* Las primeras 50 llamadas no se cobran. Luego, cada llamada cuesta $0.60

La lectura de datos se termina cuando duración es -1.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Llamadas
    Tipo,Minutos,Local Es Entero;
    Cost,TotPagar Es Real;
    Local=0;
    Escribir "Tipo de llamada Internacional=1, Nacional=2, Local=3";
    Leer Tipo;
    Escribir "Cuatos minutos";
    Leer Minutos;
    Mientras_____________
        Segun Tipo hacer
            Caso 1:
                Si Minutos >_____
                    Costo=7.59+           *3.03;
                SiNo
                    Costo=7.59;
                FinSi
            Caso 2:
                Si Minutos > 3
                    Costo=_____________________;
                SiNo
                    Costo=1.20;
                FinSi
            Caso 3:
                Local=Local+1;
                Si _________
                    Costo=.60
                SiNo
                    Costo=0;
                FinSi
        FinSegun
        TotPagar=_____________;
        Escribir "Tipo de llamada Internacional=1, Nacional=2, Local=3";
        Leer_____;
        Escribir "Cuatos minutos, para terminar minutos =-1";
        Leer ________;
    FinMientras
    Escribir "Total a pagar es ",________;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.

c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 6**: En una bodega se tiene información sobre las cantidades producidas de cada tipo de vino, a lo largo de los últimos años. Se requiere calcular e imprima lo siguiente:

- El total producido de cada tipo de vino (son 5 tipos) a lo largo de los N años.
- El total producido de vino por año.
- Año en que se produjo la mayor cantidad de litros de vino del tipo 2. Imprimir también la cantidad de litros.
- Verificar si hubo algún año en el cual no se produjo el vino tipo 3. Si existe dicho año, imprimirlo.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

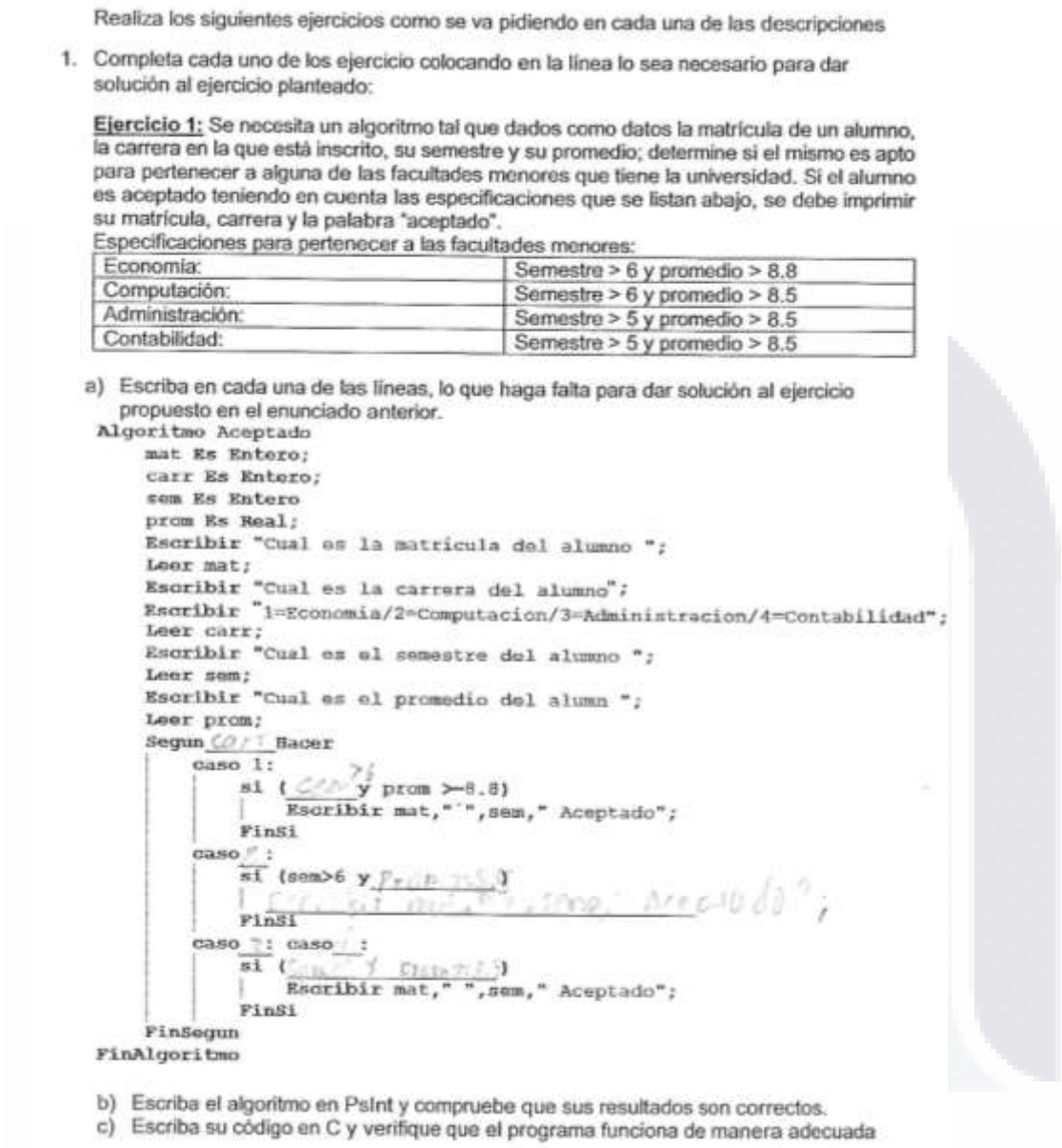
```
Algoritmo Vinos
    i,anio,j,N Es Entero;
    T1,T2,T3,T4,T5,MayT2,TotVin,vino Es Real;
    T1=0;T2=0;T3=0;T4=0;T5=0;MayT2=0;
    Escribir "Cuantos años se van a evaluar ";
    Leer N;
    para i desde __hasta N Hacer
        TotVin=__;
        para j desde 1 hasta___Hacer
            Escribir "Litros de vino del año ",i," Del tipo ",j;
            Leer ____;
            TotVin=TotVin+____;
            Segun j hacer
                Caso 1:
                    T1=__+vino;
                Caso 2:
                    T2=T2+vino;
                    Si vino>___
                        MayT2=____;
                        anio=__;
                    FinSi
                Caso 3:
                    T3=T3+____;
                    Si vino=0
                        Escribir "En el año ",i," No se produjo vino del tipo 3";
                    FinSi
                Caso 4:
                    T4=T4+vino;
                Caso 5:
                    _____;
            FinSegun
        FinPara
        Escribir "Total de litros producidos en el año ",i," son ",TotVin;
    FinPara
    Escribir "Total producido de vino tipo 1 son ",__;
    Escribir "Total producido de vino tipo 2 son ",T2;
    Escribir "Total producido de vino tipo 3 son ",T3;
    Escribir "Total producido de vino tipo 4 son ",T4;
    Escribir "Total producido de vino tipo 5 son ",T5;
    Escribir "El año en el que se produjo mayor catidad del vino tipo 2 es ",____;
    Escribir "Y se produjeron ",MayT2," Litros";
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

NOTA: al terminar, subir al Moodle los ejercicios resueltos en PseInt y los códigos de C.

## Anexo 2: Ejercicios de autocompletar resueltos por estudiantes.

Las siguientes imágenes son una muestra de material por completar que se les aplico a los alumnos del grupo experimental como parte de tratamiento mencionado en esta tesis.



Realiza los siguientes ejercicios como se va pidiendo en cada una de las descripciones

1. Completa cada uno de los ejercicio colocando en la línea lo sea necesario para dar solución al ejercicio planteado:

**Ejercicio 1:** Se necesita un algoritmo tal que dados como datos la matrícula de un alumno, la carrera en la que está inscrito, su semestre y su promedio; determine si el mismo es apto para pertenecer a alguna de las facultades menores que tiene la universidad. Si el alumno es aceptado teniendo en cuenta las especificaciones que se listan abajo, se debe imprimir su matrícula, carrera y la palabra "aceptado".

Especificaciones para pertenecer a las facultades menores:

| | |
|---|---|
| Economía: | Semestre > 6 y promedio > 8.8 |
| Computación: | Semestre > 6 y promedio > 8.5 |
| Administración: | Semestre > 5 y promedio > 8.5 |
| Contabilidad: | Semestre > 5 y promedio > 8.5 |

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Aceptado
    mat Es Entero;
    carr Es Entero;
    sem Es Entero
    prom Es Real;
    Escribir "Cual es la matricula del alumno ";
    Leer mat;
    Escribir "Cual es la carrera del alumno";
    Escribir "1=Economia/2=Computacion/3=Administracion/4=Contabilidad";
    Leer carr;
    Escribir "Cual es el semestre del alumno ";
    Leer sem;
    Escribir "Cual es el promedio del alumn ";
    Leer prom;
    Segun carr Hacer
        caso 1:
            si ( carr y prom >=8.8)
                Escribir mat,"",sem," Aceptado";
            FinSi
        caso 2 :
            si (sem>6 y prom>8.5)
                Escribir mat,"",sem," Aceptado";
            FinSi
        caso 3: caso 4 :
            si (carr = 3 y carr = 4)
                Escribir mat," ",sem," Aceptado";
            FinSi
    FinSegun
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 2:** En un hospital se ha hecho un estudio sobre los pacientes registrados durante los últimos 10 años, con el objeto de hacer una aproximación de los costos de internación por paciente. Se obtuvo un costo promedio diario según el tipo de enfermedad que aqueja al paciente. Además se pudo determinar que en promedio todos los pacientes con edad entre 14 y 22 años implican un costo adicional del 10%. La siguiente tabla expresa los costos diarios, según el tipo de enfermedad.

| Tabla | |
|---|---|
| Tipo de Enfermedad | Costo: paciente/día |
| 1 | 35 |
| 2 | 16 |
| 3 | 20 |
| 4 | 32 |

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Hospital
    TipoEnf Es Entero;
    Edad Es Entero;
    Dias Es Entero;
    Costo Es Real;
    Escribir "Tipo de enfermedad 1/2/3/4 ";
    Leer TipoEnf;
    Escribir "Edad del paciente ";
    Leer Edad;
    Escribir "Dias de hospital ";
    Leer Dias;
    Segun TipoEnf Hacer
        caso 1:
            Costo=Dias*35;
        caso 2:
            Costo=Dias*16;
        caso 3:
            Costo=Dias *20;
        caso 4:
            Costo=Dias*32;
    FinSegun
    Si (Edad > 14  y Edad <22 )
        Costo=Costo* 1,10 ;
    FinSi
    Escribir "El costo total del paciente es: ", Costo;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 3:** Un vendedor ha hecho una serie de ventas y desea conocer aquellas de $200 o menos, las mayores a $200 pero inferiores a $400, y el número de ventas de $400 o superiores a tal cantidad. Haga un diagrama de flujo que le proporcione al vendedor esta información después de haber leído los datos de entrada.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Ventas
    N,i,venta,vmenor200,ventre200y400,vmas400 Es Entero;
    vmenor200=0;
    ventre200y400=0;
    vmas400=0;
    Escribir "Cuantas son el total de ventas a leer ";
    Leer N;
    para i desde 1 hasta N Hacer
        Escribir "Cual es el monto de la venta ";
        Leer venta;
        Si (venta <=200)
            vmenor200=vmenor200+1;
        SiNo
            Si (venta <400)
                ventre200y400=ventre200y400+1;
            SiNo
                vmas400=vmas400+1;
            FinSi
        FinSi
    FinPara
    Escribir "Ventas de menos de 200 son: ",vmenor200;
    Escribir "Ventas entre 200 y 400 son: ",ventre200y400;
    Escribir "Ventas mayores de 400 son: ",vmas400;
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 4:** Supóngase que en una reciente elección hubo cuatro candidatos (con identificadores 1, 2, 3, 4). Usted habrá de encontrar, mediante un programa, el número de votos correspondiente a cada candidato y el porcentaje que obtuvo respecto al total de los votantes. El usuario tecleará los votos de manera desorganizada, tal y como se obtuvieron en la elección, el final de datos está representado por un cero.

Observe, como ejemplo, la siguiente lista:

1 3 1 4 2 2 1 4 1 1 1 2 1 3 1 4 0

Donde 1 representa un voto para el candidato 1; 3 un voto para el candidato 3; y así sucesivamente.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo votos
    C1,C2,C3,C4,totVotos,voto Es Entero;
    porm1,prom2,prom3,prom4 Es Real;
    C1=0;
    C2=0;
    C3=0;
    C4=0;
    Repetir
        Escribir "Voto para cadidato 1/2/3/4 0=Terminar";
        Leer voto;
        Segun voto hacer
            caso 1:
                C1=C1+1;
            caso 2:
                C2=C2+1;
            caso 3:
                C3=C3+1;
            caso 4:
                C4=C4+1;
        FinSegun
    Hasta Que voto=0;
    totVotos=C1+C2+C3+C4;
    prom1=C1/totVotos*100;
    prom2=C2/totVotos*100;
    prom3=C3/totVotos*100;
    Escribir "Total de votos para candiato 1-> ",C1," Porcentaje ",prom1,"%";
    Escribir "Total de votos para candiato 2-> ",C2," Porcentaje ",prom2,"%";
    Escribir "Total de votos para candiato 3-> ",C3," Porcentaje ",prom3,"%";
    Escribir "Total de votos para candiato 4-> ",C4," Porcentaje ",prom4,"%";
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

**Ejercicio 5:** Se requiere saber lo que hay que pagar por un conjunto de llamadas telefónicas. Por cada llamada se ingresa el tipo (Internacional=1, Nacional=2, Local=3) y la duración en minutos. El criterio que se sigue para calcular el costo de cada llamada es el siguiente:

*Internacional:* 3 primeros minutos $7.59 Cada minuto adicional $3.03
*Nacional: 3* primeros minutos $1.20 Cada minuto adicional $0.48
*Local:* Las primeras 50 llamadas no se cobran. Luego, cada llamada cuesta $0.60

La lectura de datos se termina cuando duración es -1.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Llamadas
    Tipo,Minutos,Local Es Entero;
    Cost,TotPagar Es Real;
    Local=0;
    Escribir "Tipo de llamada Internacional=1, Nacional=2, Local=3";
    Leer Tipo;
    Escribir "Cuatos minutos";
    Leer Minutos;
    Mientras
    |   Segun Tipo hacer
    |   |   Caso 1:
    |   |   |   Si Minutos >
    |   |   |   |   Costo=7.59+          *3.03;
    |   |   |   SiNo
    |   |   |   |   Costo=7.59;
    |   |   |   FinSi
    |   |   Caso 2:
    |   |   |   Si Minutos > 3
    |   |   |   |   Costo=
    |   |   |   SiNo
    |   |   |   |   Costo=1.20;
    |   |   |   FinSi
    |   |   Caso 3:
    |   |   |   Local=Local+1;
    |   |   |   Si
    |   |   |   |   Costo=.60
    |   |   |   SiNo
    |   |   |   |   Costo=0;
    |   |   |   FinSi
    |   FinSegun
    |   TotPagar=          ;
    |   Escribir "Tipo de llamada Internacional=1, Nacional=2, Local=3";
    |   Leer          ;
    |   Escribir "Cuatos minutos, para terminar minutos =-1";
    |   Leer          ;
    FinMientras
    Escribir "Total a pagar es ",
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

Ejercicio 6: En una bodega se tiene información sobre las cantidades producidas de cada tipo de vino, a lo largo de los últimos años. Se requiere calcular e imprima lo siguiente:

- El total producido de cada tipo de vino (son 5 tipos) a lo largo de los N años.
- El total producido de vino por año.
- Año en que se produjo la mayor cantidad de litros de vino del tipo 2. Imprimir también la cantidad de litros.
- Verificar si hubo algún año en el cual no se produjo el vino tipo 3. Si existe dicho año, imprimirlo.

a) Escriba en cada una de las líneas, lo que haga falta para dar solución al ejercicio propuesto en el enunciado anterior.

```
Algoritmo Vinos
    i,anio,j,N Es Entero;
    T1,T2,T3,T4,T5,MayT2,TotVin,vino Es Real;
    T1=0;T2=0;T3=0;T4=0;T5=0;MayT2=0;
    Escribir "Cuantos años se van a evaluar ";
    Leer N;
    para i desde 1 hasta N Hacer
        TotVin=0;
        para j desde 1 hasta 5 Hacer
            Escribir "Litros de vino del año ",i," Del tipo ",j;
            Leer vino;
            TotVin=TotVin+vino;
            Segun j hacer
                Caso 1:
                    T1=T1+vino;
                Caso 2:
                    T2=T2+vino;
                    Si vino>MayT2
                        MayT2=vino;
                        anio=i;
                    FinSi
                Caso 3:
                    T3=T3+vino;
                    Si vino=0
                        Escribir "En el año ",i," No se produjo vino del tipo 3";
                    FinSi
                Caso 4:
                    T4=T4+vino;
                Caso 5:
                    T5=T5+vino;
            FinSegun
        FinPara
        Escribir "Total de litros producidos en el año ",i," son ",TotVin;
    FinPara
    Escribir "Total producido de vino tipo 1 son ",T1;
    Escribir "Total producido de vino tipo 2 son ",T2;
    Escribir "Total producido de vino tipo 3 son ",T3;
    Escribir "Total producido de vino tipo 4 son ",T4;
    Escribir "Total producido de vino tipo 5 son ",T5;
    Escribir "El año en el que se produjo mayor catidad del vino tipo 2 es ",anio;
    Escribir "Y se produjeron ",MayT2," Litros";
FinAlgoritmo
```

b) Escriba el algoritmo en PsInt y compruebe que sus resultados son correctos.
c) Escriba su código en C y verifique que el programa funciona de manera adecuada

NOTA: al terminar, subir al Moodle los ejercicios resueltos en PseInt y los códigos de C.

**Anexo 3: Ejercicios contestados herramienta PSeInt**

Los siguientes son una muestra de los ejercicios contestados por los estudiantes del grupo experimental en la herramienta PSeInt, mismo que fueron entregados vía plataforma Moodle.

## SUBIR ALGORITMOS REALIZADOS EN PSeint Lunes 13 noviembre

SUBIR ALGORITMOS REALIZADOS EN PSeint Lunes 13 noviembre

Grupos separados

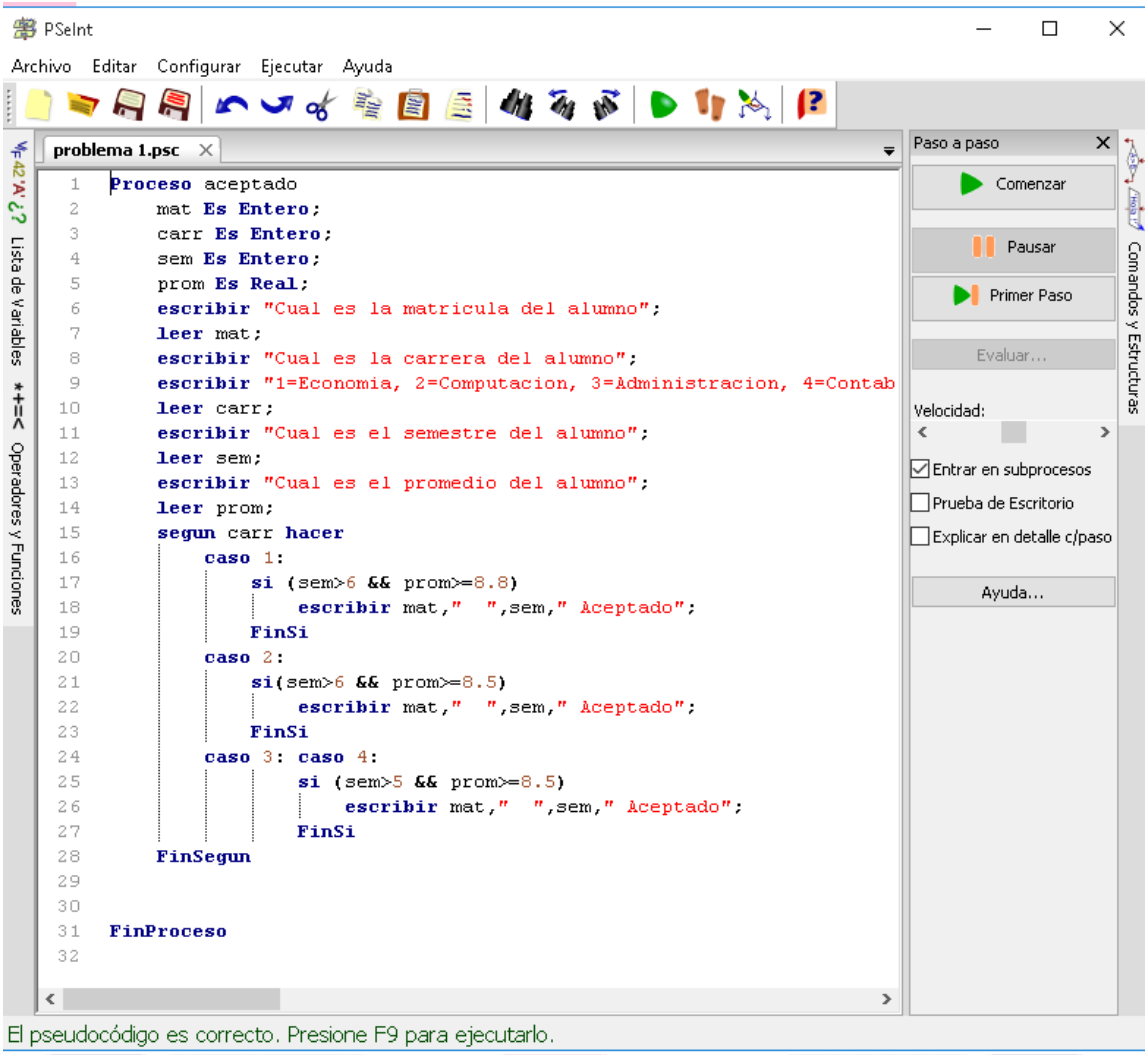
ISC_1C_AGO_DIC_2017

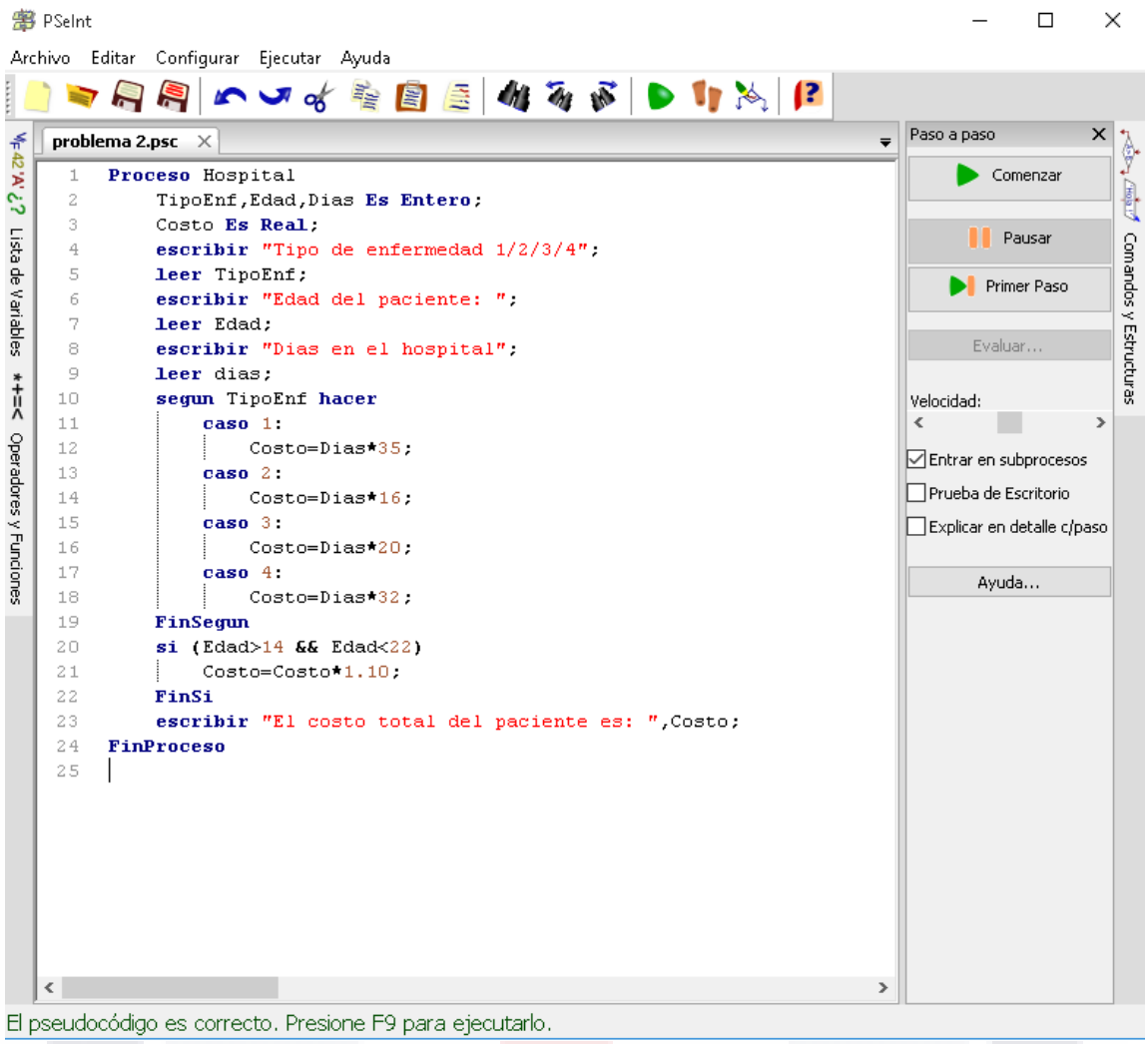
### Sumario de calificaciones

| | |
|---|---|
| Participantes | 49 |
| Enviados | 35 |
| Necesita calificarse | 35 |
| Fecha de entrega | lunes, 13 de noviembre de 2017, 23:45 |

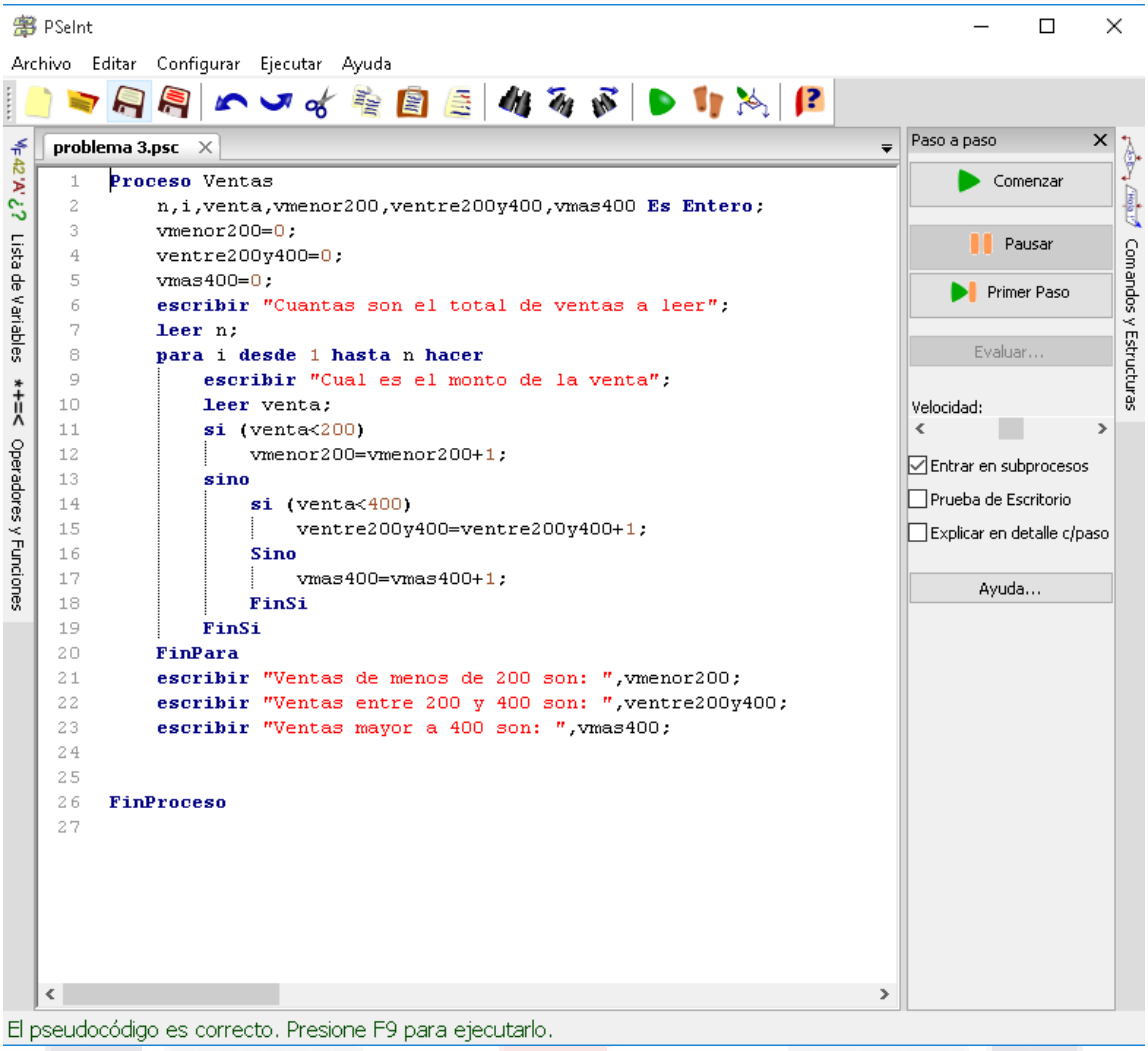
Página: 1 2 3 4 (Siguiente)

| Seleccionar | Imagen del usuario | Nombre / Apellido(s) | Estatus | Última modificación (entrega) | Envíos de archivo |
|---|---|---|---|---|---|
| ☐ | | Ruben Marquez Ruiz | Enviado para calificar | sábado, 11 de noviembre de 2017, 16:57 | problemas pseint.zip |
| ☐ | | Juan Pablo Quintanilla Gonzalez | Enviado para calificar | sábado, 11 de noviembre de 2017, 18:26 | PseInt.rar |
| ☐ | | Víctor Adrián Soto Dávila | Enviado para calificar 1 día 21 horas después | miércoles, 15 de noviembre de 2017, 20:58 | programasMaestraBlanca.rar |
| ☐ | | Jose Rodrigo Martinez Solis | Enviado para calificar | lunes, 13 de noviembre de 2017, 23:29 | Algoritmos.rar |
| ☐ | | JUAN CARLOS Alvarez | Enviado para calificar | martes, 14 de noviembre de 2017, 15:47 | progrmaspseint.rar |

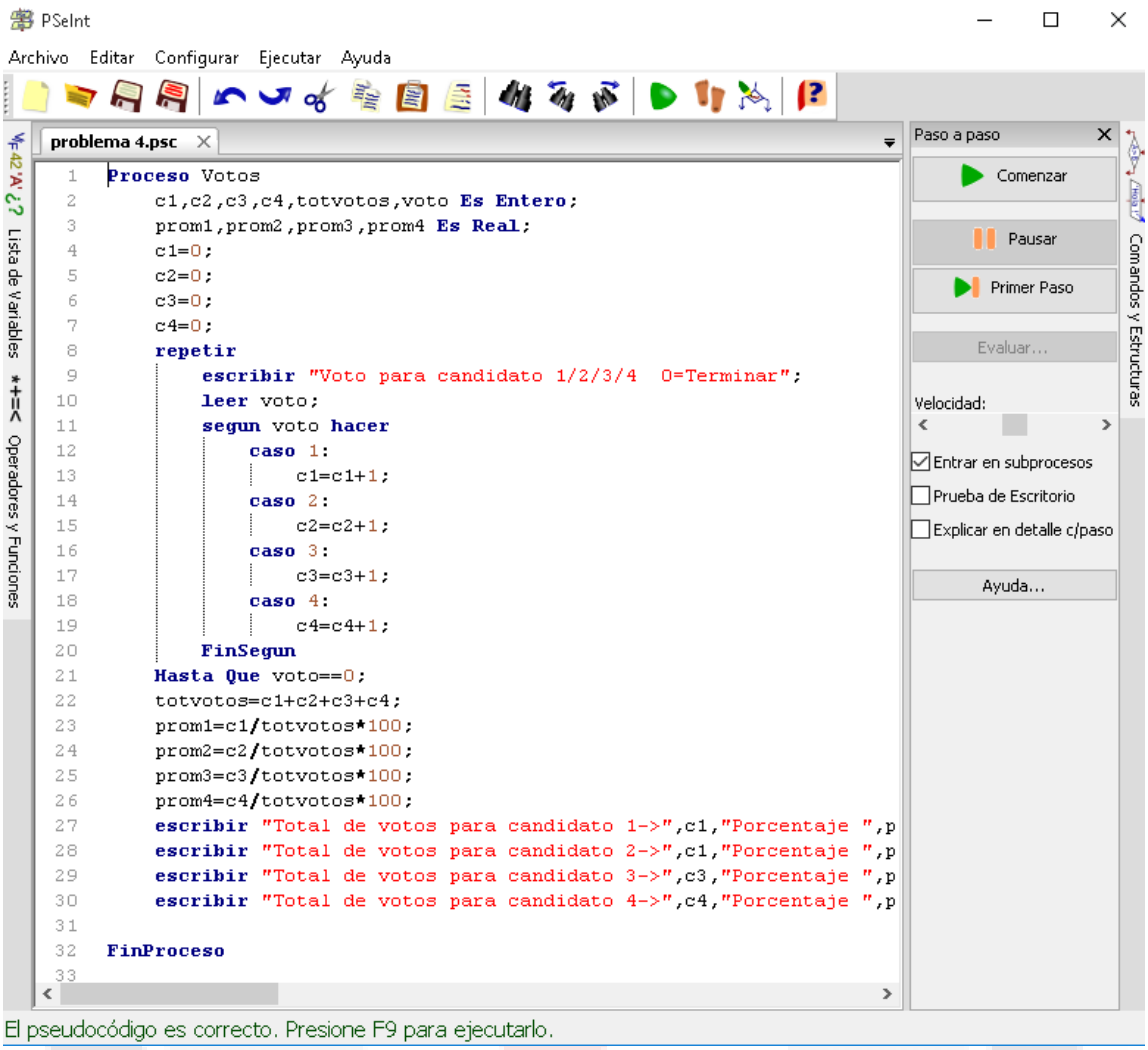




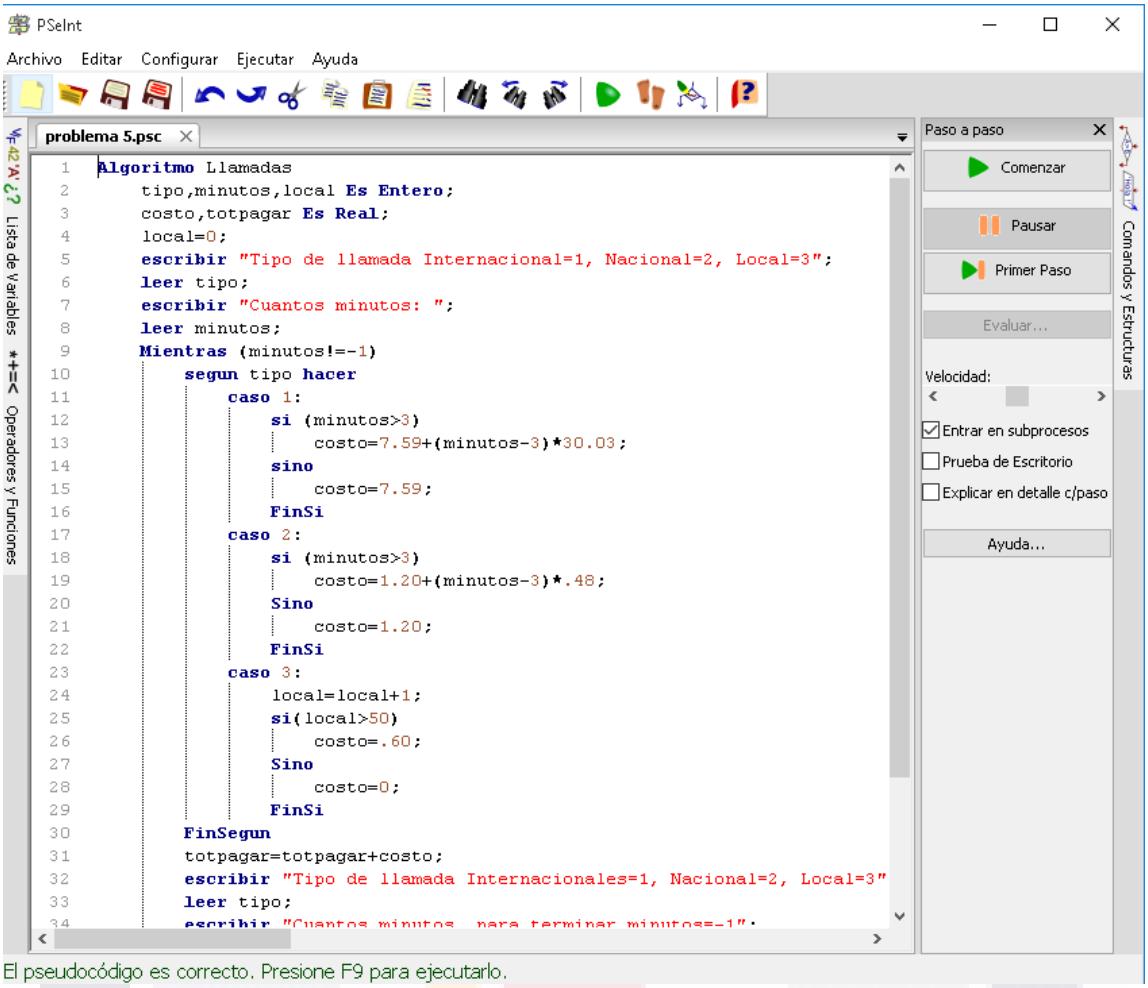

```
1  Proceso Hospital
2      TipoEnf,Edad,Dias Es Entero;
3      Costo Es Real;
4      escribir "Tipo de enfermedad 1/2/3/4";
5      leer TipoEnf;
6      escribir "Edad del paciente: ";
7      leer Edad;
8      escribir "Dias en el hospital";
9      leer dias;
10     segun TipoEnf hacer
11         caso 1:
12             Costo=Dias*35;
13         caso 2:
14             Costo=Dias*16;
15         caso 3:
16             Costo=Dias*20;
17         caso 4:
18             Costo=Dias*32;
19     FinSegun
20     si (Edad>14 && Edad<22)
21         Costo=Costo*1.10;
22     FinSi
23     escribir "El costo total del paciente es: ",Costo;
24 FinProceso
25
```



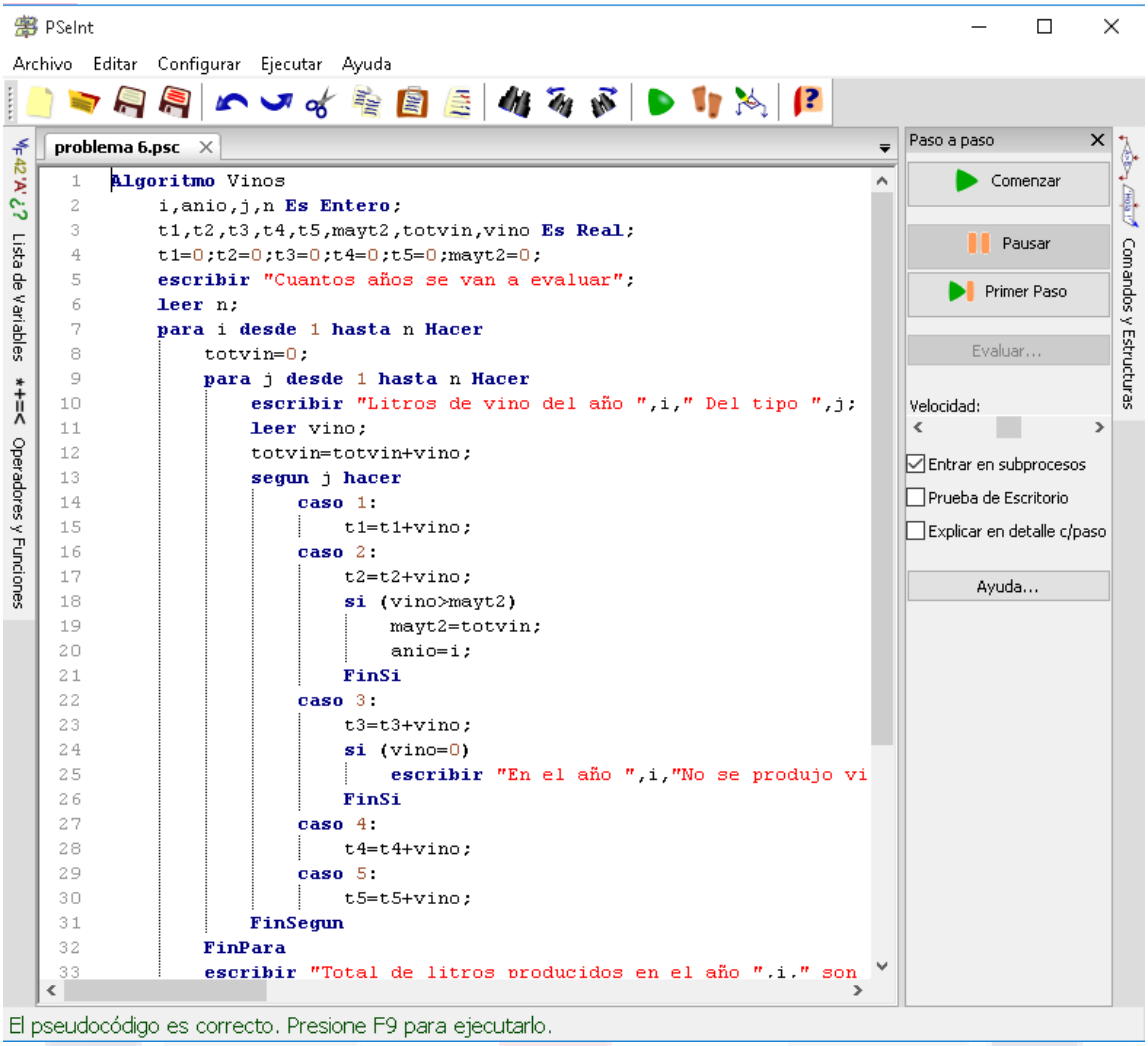

```
1  Proceso Ventas
2      n,i,venta,vmenor200,ventre200y400,vmas400 Es Entero;
3      vmenor200=0;
4      ventre200y400=0;
5      vmas400=0;
6      escribir "Cuantas son el total de ventas a leer";
7      leer n;
8      para i desde 1 hasta n hacer
9          escribir "Cual es el monto de la venta";
10         leer venta;
11         si (venta<200)
12             vmenor200=vmenor200+1;
13         sino
14             si (venta<400)
15                 ventre200y400=ventre200y400+1;
16             Sino
17                 vmas400=vmas400+1;
18             FinSi
19         FinSi
20     FinPara
21     escribir "Ventas de menos de 200 son: ",vmenor200;
22     escribir "Ventas entre 200 y 400 son: ",ventre200y400;
23     escribir "Ventas mayor a 400 son: ",vmas400;
24
25
26 FinProceso
27
```




```
1  Proceso Votos
2      c1,c2,c3,c4,totvotos,voto Es Entero;
3      prom1,prom2,prom3,prom4 Es Real;
4      c1=0;
5      c2=0;
6      c3=0;
7      c4=0;
8      repetir
9          escribir "Voto para candidato 1/2/3/4  0=Terminar";
10         leer voto;
11         segun voto hacer
12             caso 1:
13                 c1=c1+1;
14             caso 2:
15                 c2=c2+1;
16             caso 3:
17                 c3=c3+1;
18             caso 4:
19                 c4=c4+1;
20         FinSegun
21     Hasta Que voto==0;
22     totvotos=c1+c2+c3+c4;
23     prom1=c1/totvotos*100;
24     prom2=c2/totvotos*100;
25     prom3=c3/totvotos*100;
26     prom4=c4/totvotos*100;
27     escribir "Total de votos para candidato 1->",c1,"Porcentaje ",p
28     escribir "Total de votos para candidato 2->",c1,"Porcentaje ",p
29     escribir "Total de votos para candidato 3->",c3,"Porcentaje ",p
30     escribir "Total de votos para candidato 4->",c4,"Porcentaje ",p
31
32 FinProceso
33
```




```
1  Algoritmo Llamadas
2      tipo,minutos,local Es Entero;
3      costo,totpagar Es Real;
4      local=0;
5      escribir "Tipo de llamada Internacional=1, Nacional=2, Local=3";
6      leer tipo;
7      escribir "Cuantos minutos: ";
8      leer minutos;
9      Mientras (minutos!=-1)
10         segun tipo hacer
11             caso 1:
12                 si (minutos>3)
13                     costo=7.59+(minutos-3)*30.03;
14                 sino
15                     costo=7.59;
16                 FinSi
17             caso 2:
18                 si (minutos>3)
19                     costo=1.20+(minutos-3)*.48;
20                 Sino
21                     costo=1.20;
22                 FinSi
23             caso 3:
24                 local=local+1;
25                 si(local>50)
26                     costo=.60;
27                 Sino
28                     costo=0;
29                 FinSi
30         FinSegun
31         totpagar=totpagar+costo;
32         escribir "Tipo de llamada Internacionales=1, Nacional=2, Local=3"
33         leer tipo;
34         escribir "Cuantos minutos  para terminar minutos=-1";
```



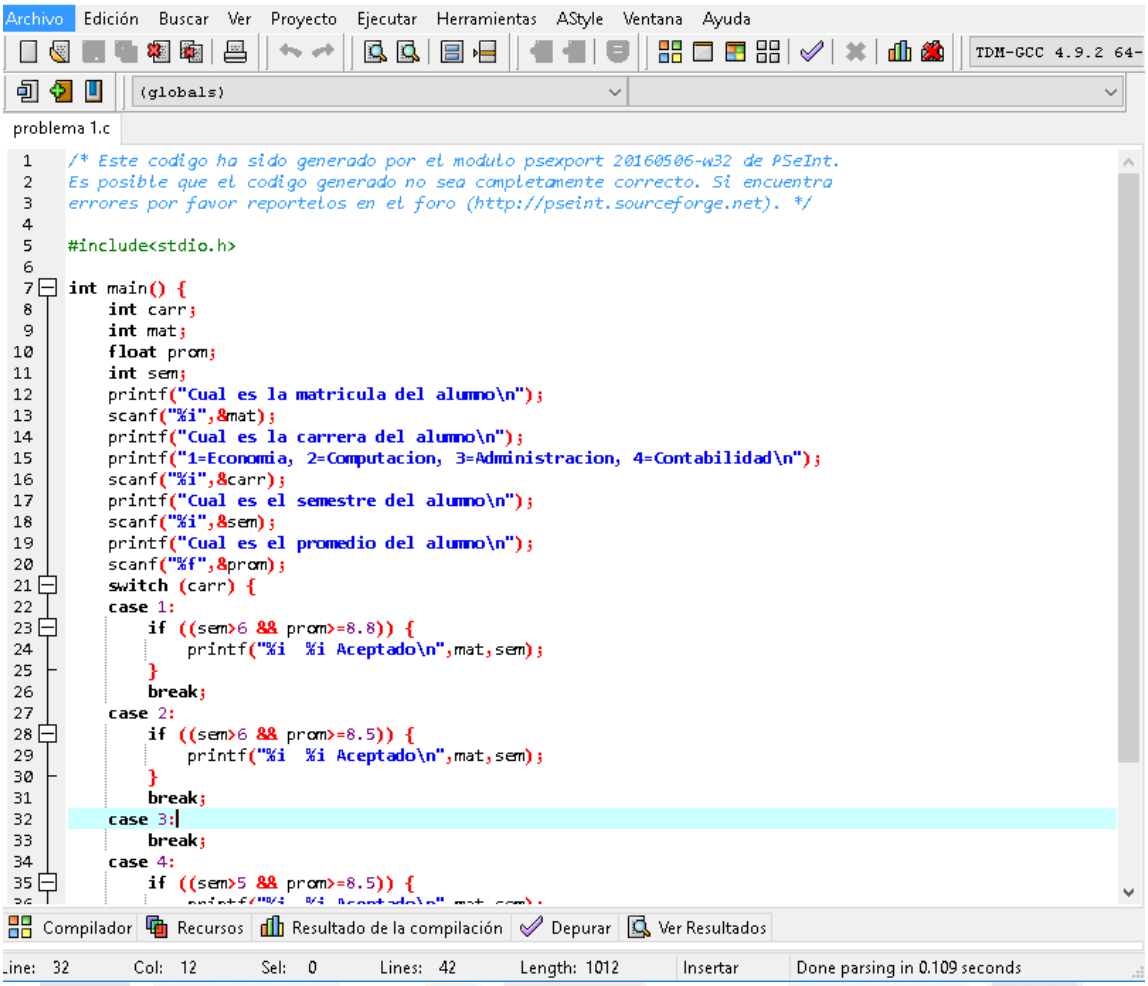

```
1  Algoritmo Vinos
2      i,anio,j,n Es Entero;
3      t1,t2,t3,t4,t5,mayt2,totvin,vino Es Real;
4      t1=0;t2=0;t3=0;t4=0;t5=0;mayt2=0;
5      escribir "Cuantos años se van a evaluar";
6      leer n;
7      para i desde 1 hasta n Hacer
8          totvin=0;
9          para j desde 1 hasta n Hacer
10             escribir "Litros de vino del año ",i," Del tipo ",j;
11             leer vino;
12             totvin=totvin+vino;
13             segun j hacer
14                 caso 1:
15                     t1=t1+vino;
16                 caso 2:
17                     t2=t2+vino;
18                     si (vino>mayt2)
19                         mayt2=totvin;
20                         anio=i;
21                     FinSi
22                 caso 3:
23                     t3=t3+vino;
24                     si (vino=0)
25                         escribir "En el año ",i,"No se produjo vi
26                     FinSi
27                 caso 4:
28                     t4=t4+vino;
29                 caso 5:
30                     t5=t5+vino;
31             FinSegun
32         FinPara
33         escribir "Total de litros producidos en el año ",i," son
```

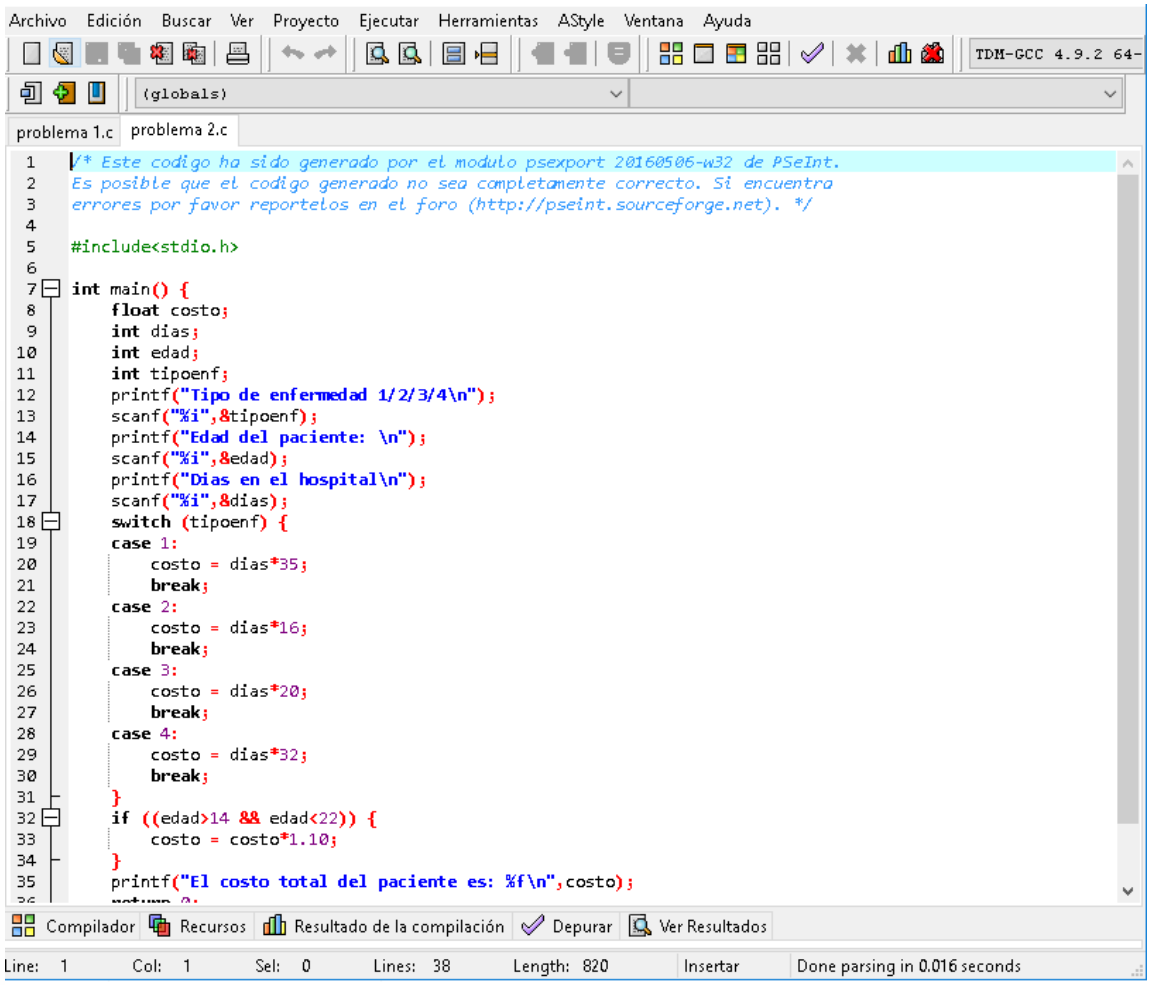
**Anexo 4: Ejercicios programados en C/C++**

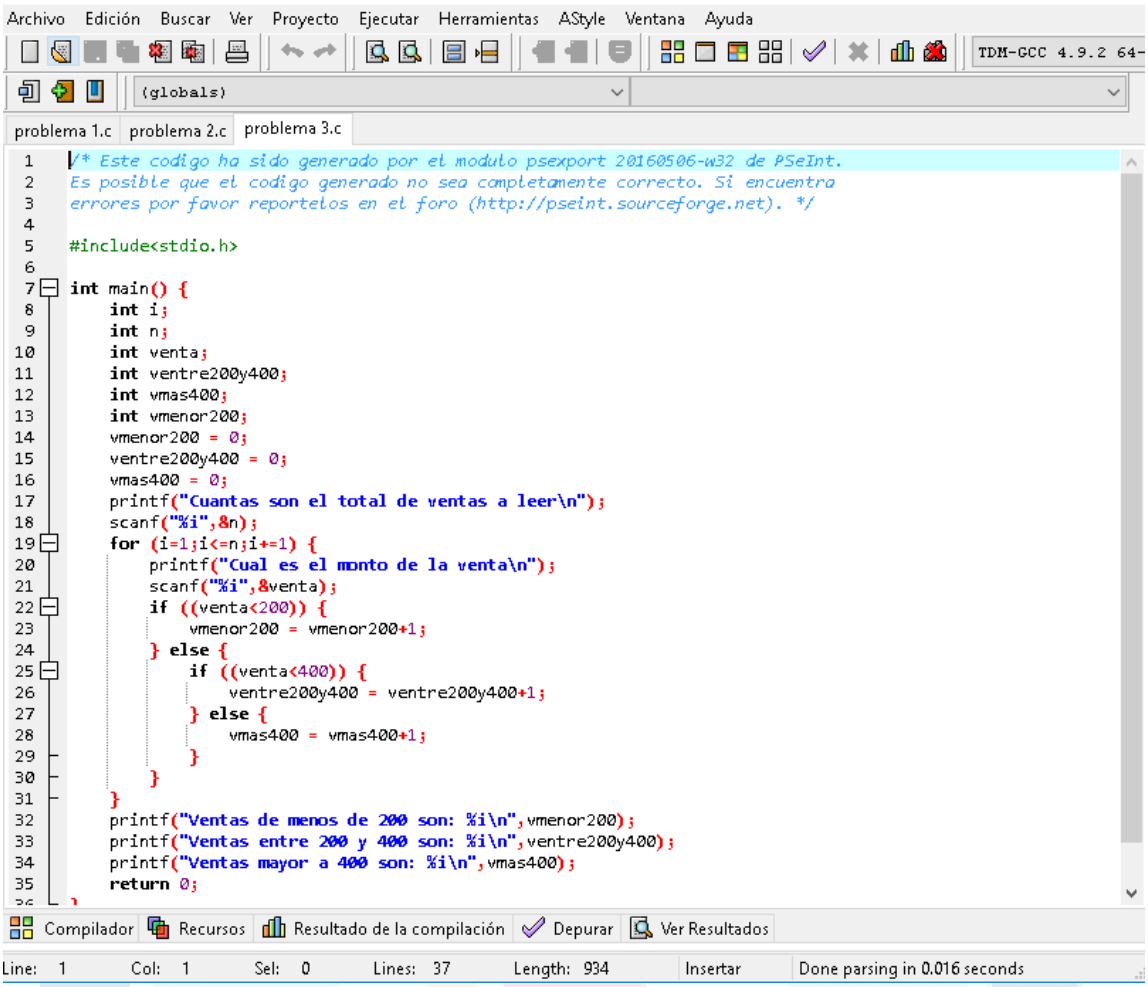
Los siguientes son una muestra de los ejercicios contestados por los estudiantes del grupo experimental en la herramienta PSeInt y posteriormente codificados en el lenguaje de programación c/c++, mismo que fueron entregados vía plataforma Moodle

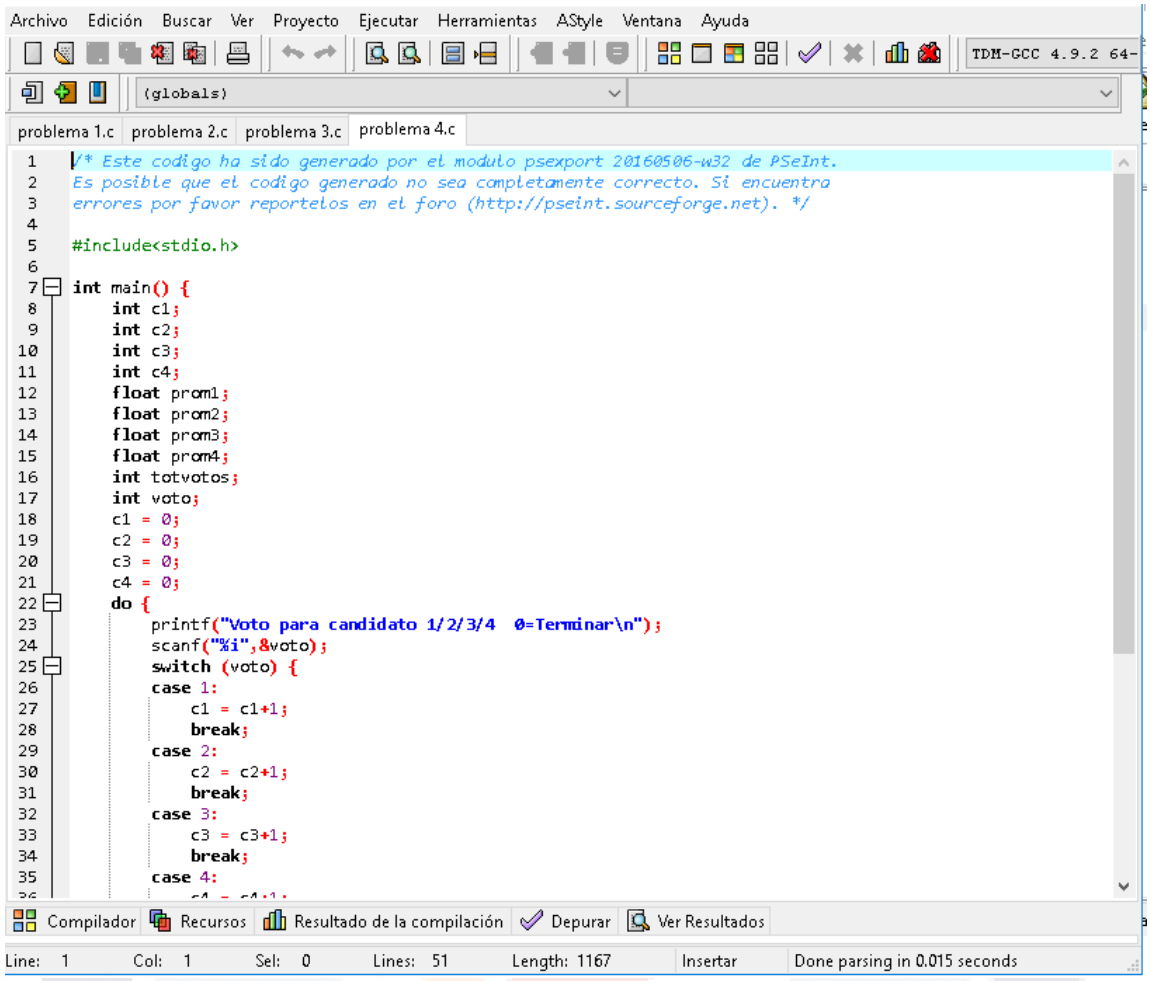
Página: 1 2 3 4 (Siguiente)

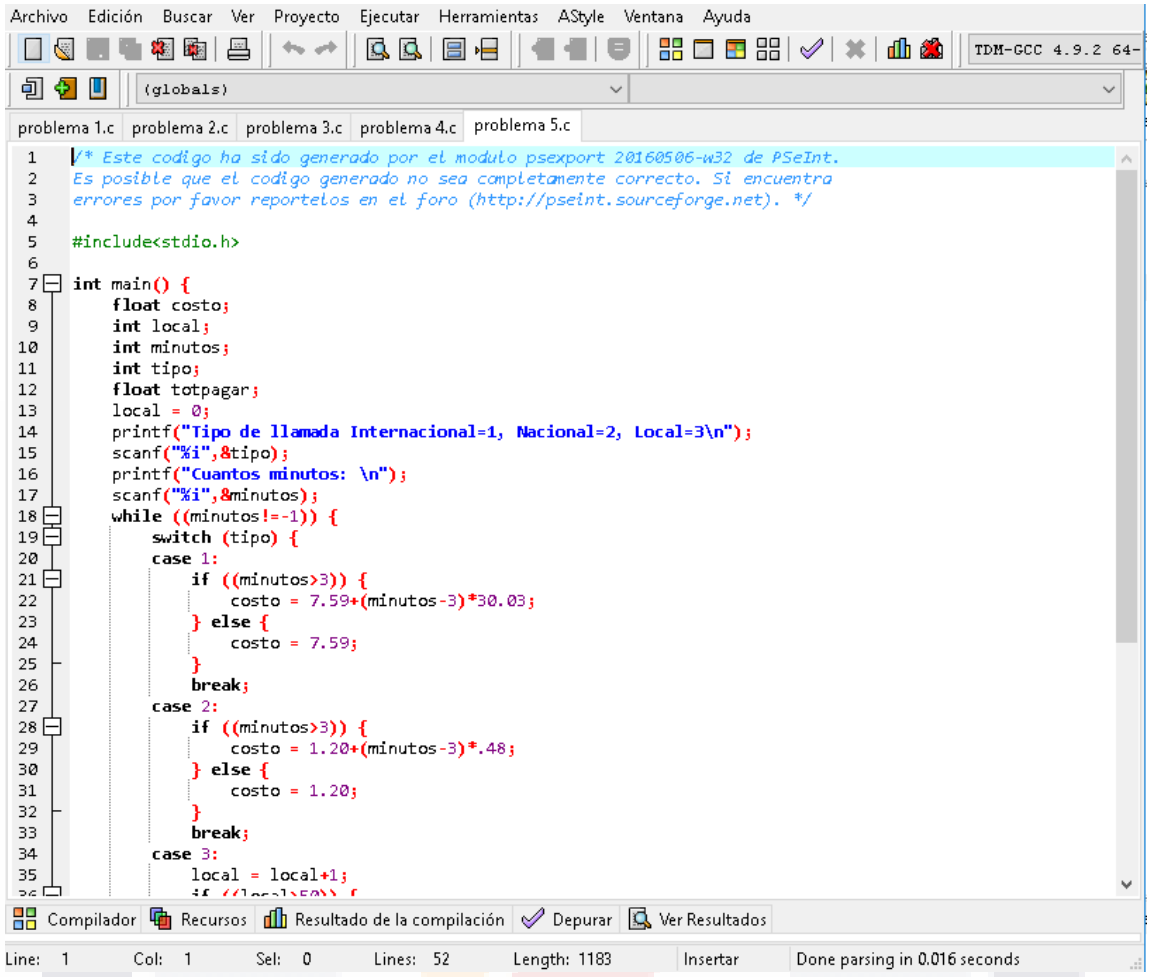
| Seleccionar | Imagen del usuario | Nombre / Apellido(s) | Estatus | Última modificación (entrega) | Envíos de archivo |
|---|---|---|---|---|---|
| ☐ | | Ruben Marquez Ruiz | Enviado para calificar | lunes, 13 de noviembre de 2017, 21:50 | problemas en c.zip |
| ☐ | | Juan Pablo Quintanilla Gonzalez | Enviado para calificar | martes, 14 de noviembre de 2017, 19:00 | PseInt.rar |
| ☐ | | Víctor Adrián Soto Dávila | Enviado para calificar 21 horas 48 minutos después | miércoles, 15 de noviembre de 2017, 21:33 | pseint a C.rar |
| ☐ | | Jose Rodrigo Martinez Solis | Enviado para calificar | lunes, 13 de noviembre de 2017, 23:23 | pseint.rar |
| ☐ | | JUAN CARLOS Alvarez Martinez | Enviado para calificar | martes, 14 de noviembre de 2017, 15:48 | programasc.rar |

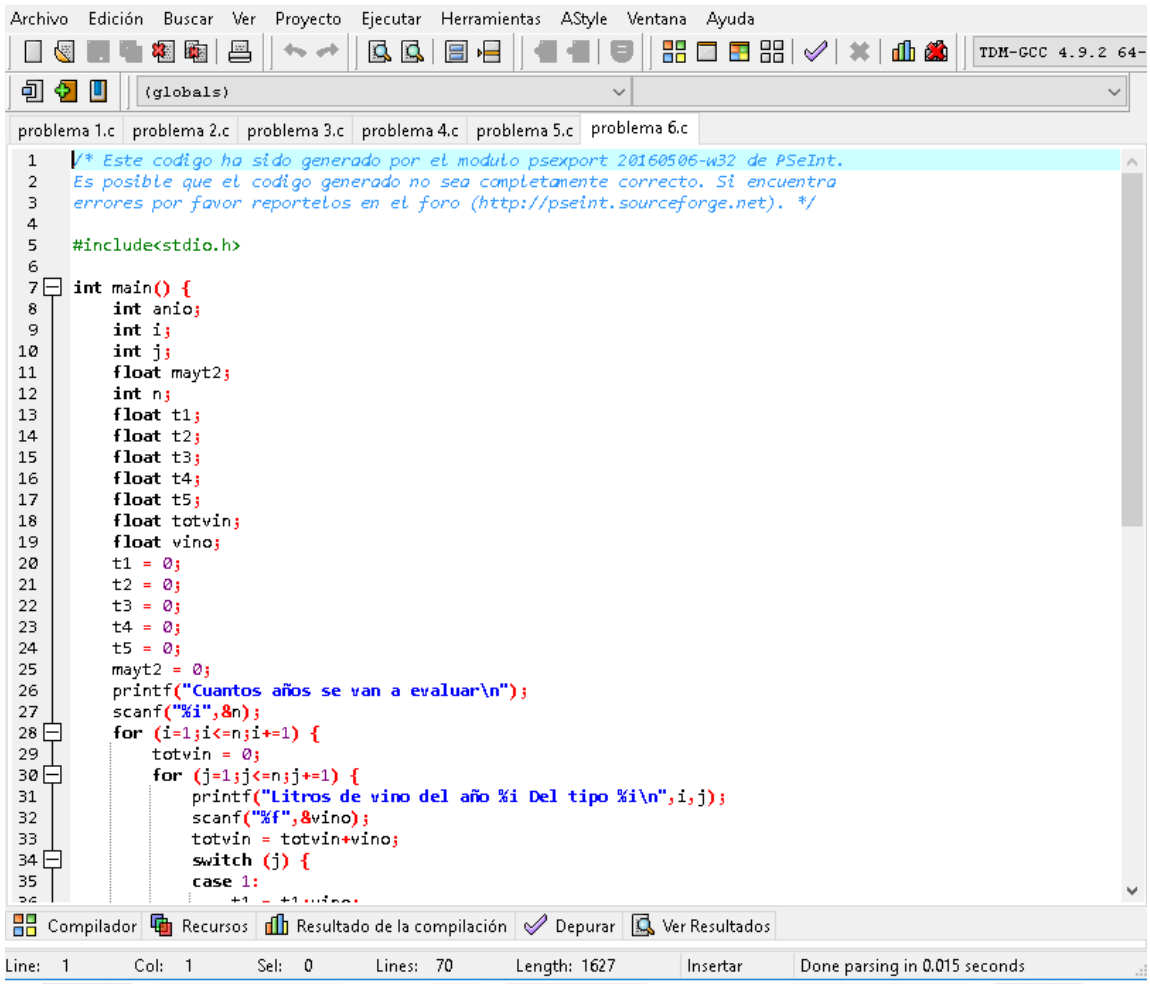












**Anexo 5: Bachilleratos de origen**

A continuación, se muestra una relación con los bachilleratos de origen de los estudiantes del grupo experimental y del grupo de control.

## Grupo de control

| Id | Nombre | Bachillerato de Origen |
|---|---|---|
| 239935 | AGUAYO GARZA MARCO ANTONIO | C.E.C.Y.T.E.A. PLANTEL RINCON DE ROMOS |
| 176410 | ALVARADO ORTIZ LEONARDO MARTIN | C.B.T.I.S. NO. 168 |
| 242249 | ARROYO VALDIVIA DANIEL TZOALI | C.B.T.I.S. NO. 168 |
| 174374 | AVILA HERMOSILLO LUIS OMAR | BACHILLERATO DE LA U.A.A. |
| 112795 | CALDERON BAÑALES ALEX IVAN | BACHILLERATO ABIERTO (AGUASCALIENTES) |
| 175077 | CAMARGO MORALES VICTOR ALEJANDRO | C.B.T.I.S. NO. 168 |
| 224181 | CARREON RUIZ ALAN JOSUE | CENTRO DE ESTUDIOS BACH. JESUS REYES HEROLES |
| 189674 | CARRILLO VILLALPANDO RICARDO | C.B.T.I.S. NO. 168 |
| 234640 | CORTES PEREZ DAVID FRANCISCO | C.B.T.I.S. NO. 168 |
| 241828 | ESCOBAR TENTLE AXEL ISAI | C.E.C.Y.T.E.A. PLANTEL SAN JOSE DE GRACIA |
| 188129 | FIGUEROA GARCIA ERIC | BACHILLERATO DE LA U.A.A. |
| 237790 | FLORES FLORES ALEXIA PILAR | C.B.T.A. NO. 61 |
| 235664 | FLORES STIKER SHARE NICOLE | ESCUELA NORMAL DE AGUASCALIENTES |
| 238298 | GARCIA BERUMEN ALVIN | INSTITUTO VICTOR MANUEL CASTELAZO MURIEL |
| 242233 | GARCIA GUTIERREZ CARLOS | C.B.T.I.S. NO. 168 |
| 206570 | GARCIA MARTINEZ KEVIN ISRAEL | C.E.T.I.S. NO. 80 |
| 204243 | GARCIA VENTURA PAOLA REBECA | CONALEP PROF. J. REFUGIO ESPARZA REYES(BIVALENTE) |
| 188067 | GONZALEZ CAMPOS CARLOS ALEJANDRO | BACHILLERATO DE LA U.A.A. |
| 235551 | GONZALEZ ESPARZA OSCAR | CENTRO EDUCATIVO "JOSE VASCONCELOS" |
| 173900 | GONZALEZ PALACIOS RICARDO | BACHILLERATO DE LA U.A.A |

| | | |
|---|---|---|
| 187974 | GONZALEZ VILLALPANDO LUIS EDUARDO | BACHILLERATO DE LA U.A.A. |
| 234829 | GUERRERO CARRERA JESSICA | C.B.T.I.S. NO. 39 |
| 198300 | GUTIERREZ SALAZAR PAOLA | INSTITUTO AGUASCALIENTES A.C. |
| 205628 | HERNANDEZ PRIETO URIEL IVAN | CENTRO DE ESTUDIOS DE BACHILLERATO |
| 189501 | HUERTA SALAS DAVID MOISES | BACHILLERATO DE LA U.A.A |
| 188539 | JIMENEZ CORNEJO LUIS JONATHAN | BACHILLERATO DE LA U.A.A. |
| 220237 | LOERA LANDEROS HENRY ALEJANDRO | C.E.C.Y.T.E.A. PLANTEL CALVILLO |
| 174517 | LOPEZ DE LA TORRE MIGUEL ANGEL | BACHILLERATO DE LA U.A.A |
| 240426 | MAAFS ATILANO RODRIGO | COLEGIO DE EST. CIENT. Y TEC.(ENCARNACION DE DIAZ) |
| 205683 | MACIAS BORJA AARON JESUS | C.B.T.A. NO. 103 |
| 208079 | MARTINEZ SALAS RAYMUNDO | C.E.C.Y.T.E.A. CIUDAD SATELITE MORELOS |
| 234743 | MEDINA MUÑOZ FRANCISCO JAVIER | C.B.T.I.S. NO. 168 |
| 206375 | MENDEZ OSORIO LUIS GIOVANNI | C.B.T.I.S. NO. 39 |
| 242135 | MONTAÑEZ CARLIN ORLANDO | C.B.T.I.S. NO. 215 (LORETO) |
| 138224 | NEQUIS CARRANZA MARCO ALEJANDRO | PREPARATORIA PARTICULAR DE CUAUTITLAN |
| 240045 | OLIVETTI ALVAREZ VICTOR HUGO | C.E.C.Y.T.E.A. PLANTEL PABELLON DE ARTEAGA |
| 242849 | PADILLA ALEXIS | PREP. PART. INC. "COLEGIO MIGUEL DE BOLONIA" |
| 174166 | PADILLA ALONSO ALFONSO | BACHILLERATO DE LA U.A.A |
| 161415 | PAVON ROMO JORGE MANUEL | COLEGIO LINCOLN, A.C |
| 240156 | PEDROZA CARRILLO JUAN CARLOS | CONALEP PROF. J. REFUGIO ESPARZA REYES(BIVALENTE) |
| 199151 | QUESADA ROMO JAVIER | ESCUELA PREPARATORIA ANGEL ANGUIANO, A.C |
| 209864 | RAMIREZ GONZALEZ EMMANUEL | C.B.T.I.S. NO. 168 |
| 206660 | RAMIREZ GONZALEZ ISAIAS EMMANUEL | C.B.T.I.S. NO. 168 |
| 238555 | ROBLEDO AGUILAR JOSE RAMON | C.B.T.I.S. NO. 215 (LORETO) |
| 64259 | RODRIGUEZ HERRERA GABRIEL EMMANUEL | ESCUELA PREPARATORIA JOSE MARIA MORELOS Y PAVON |
| 188237 | RODRIGUEZ REYES DANIELA YAEL | BACHILLERATO DE LA U.A.A |
| 199426 | RUIZ BERNAL OMAR ARTURO | PREPARATORIA ENTORNO |

| | | |
|---|---|---|
| 225777 | RUIZ DE LOERA CRISTOBAL EMMANUEL | C.B.T.A. NO. 61 |
| 238784 | SALAS LOPEZ LUIS FERNANDO | C.E.C.Y.T.E.A. PLANTEL 07 FERROCARRILES |
| 198543 | SANCHEZ SOTO EDMUNDO | BACHILLERATO DE LA U.A.A. |
| 235137 | SERNA DAVILA MARIELA TERESA | C.E.C.Y.T.E.A. PLANTEL PABELLON DE ARTEAGA |
| 239411 | SORT DE SANZ RUIZ JORGE ARMANDO | UNIVERSIDAD VASCO DE QUIROGA |
| 145079 | TAJONAR TORTOLERO JULIO HUMBERTO | ESCUELA DE LA CIUDAD DE AGUASCALIENTES, A.C |
| 238226 | TORRES DE LEON ADRIANA | C.B.T.I.S. NO. 284 |
| 175405 | WONG MARTINEZ LUIS ENRIQUE | BACHILLERATO DE LA U.A.A |
| 236667 | ZARATE ROBLES DAVID ISAI | C.B.T.I.S. NO. 168 |

10 CBTIS 168

## Grupo Experimental

| Id | Nombre | Bachillerato de origen |
|---|---|---|
| 238154 | ACERO GONZALEZ FELIPE ENRIQUE | C.B.T.I.S. NO. 168 |
| 234903 | AGUILAR REYES JOSE EDUARDO | C.B.T.I.S. NO. 39 |
| 235024 | ALVA MUÑOZ ERNESTO | C.B.T.I.S. NO. 168 |
| 237129 | ALVAREZ MARTINEZ JUAN CARLOS | CENTRO DE BACHILLERATO TECNICO EN COMPUTACION |
| 221382 | ARAIZA PERALTA MIRIAM GUADALUPE | CONALEP PLANTEL TEPEZALA (BIVALENTE) |
| 223643 | AVILA NAVARRO JOSE LUIS | ESC. PREP. "COLEGIO DEL CENTRO" |
| 237000 | AVILA SILVA HILDA ANETTE | C.E.C.Y.T.E.A. EMSAD JESUS MARIA |
| 175310 | BARBA MARTINEZ SANTIAGO | BACHILLERATO DE LA U.A.A |
| 174372 | BARRERA RANGEL KAREN JAQUELINE | BACHILLERATO DE LA U.A.A. |
| 241107 | BECERRA ZERMEÑO JOSE ABRAHAM | COLEGIO DE EST. CIENT. Y TEC.(ENCARNACION DE DIAZ |
| 240434 | DE SANTOS DE SANTOS LUIS DIEGO | COLEGIO DE EST. CIENT. Y TEC.(ENCARNACION DE DIAZ |
| 234560 | DELGADO CAMPOS CYNTHIA GUADALUPE | C.B.T.I.S. NO. 282 |

| | | |
|---|---|---|
| 234573 | DELGADO SILVA DAVID SURIE | C.B.T.I.S. NO. 282 |
| 223024 | ESCOBAR LOPEZ KELVIN | PREPARATORIA REGIONAL DE LAGOS DE MOREN |
| 220436 | FELIX MORALES WILLIAM ABNER | C.B.T.I.S. NO. 168 |
| 241666 | GARCIA VIRAMONTES LUIS ANDRES | C.E.C.Y.T.E.A. PLANTEL CALVILLO |
| 189098 | GONZALEZ JESSICA DAYANA | BACHILLERATO DE LA U.A.A |
| 189071 | GONZALEZ SALAS MIGUEL | BACHILLERATO DE LA U.A.A |
| 187898 | GOROSTIOLA MENDEZ EDUARDO | BACHILLERATO DE LA U.A.A |
| 236595 | GUTIERREZ PLASCENCIA DANIELA | C.B.T.I.S. NO. 282 |
| 234557 | HERNANDEZ HERNANDEZ JENNIFER | C.B.T.I.S. NO. 282 |
| 239919 | HERRERA CAMPOS ESTEBAN BENJAMIN | C.B.T.I.S. NO. 168 |
| 217419 | HERRERA DIAZ EDGAR ANDRES | C.B.T.I.S. NO. 39 |
| 235064 | HORTA VARGAS SONIA PAULINA | CENTRO DE EST. DE BACHILLERATO "AGUASCALIENTES |
| 174969 | JIMENEZ CEBALLOS SEBASTIAN | BACHILLERATO DE LA U.A.A |
| 238470 | LOPEZ LOPEZ ALBERTO DE JESUS | C.B.T.I.S. NO. 247 (TEOCALTICHE) |
| 174478 | LOPEZ ZAVALA DIEGO NAZARETH | BACHILLERATO DE LA U.A.A |
| 239820 | LUNA DIAZ LUIS PABLO | C.E.T.I.S. NO. 155 |
| 188036 | MARQUEZ RUIZ RUBEN | BACHILLERATO DE LA U.A.A |
| 234741 | MARTINEZ CONTRERAS FRANCISCO SAMAEL | C.B.T.I.S. NO. 168 |
| 221718 | MARTINEZ MENDEZ JESUS ALBERTO | C.B.T.I.S. NO. 39 |
| 173869 | MARTINEZ SOLIS JOSE RODRIGO | BACHILLERATO DE LA U.A.A |
| 222895 | MEDINA QUEZADA JOSE JAIME | ESC. PREP. E. ING. Y GRAL. FELIPE B. BERRIOZAB |
| 223010 | MONTOYA SILVA ACXEL FERNANDO | CENTRO DE SERV. DE EDUC. MEDIA SUPERIOR(PEDREGOSO |
| 222946 | MUÑOZ SERNA J. REFUGIO | ESC. PREP. E. ING. Y GRAL. FELIPE B. BERRIOZAB |
| 238556 | PADILLA GUZMAN JOSE DE JESUS | CENTRO DE ESTUDIOS BACH. JESUS REYES HEROLES |
| 175307 | PALOS ESTRADA MOISES ESTEBAN | BACHILLERATO DE LA U.A.A |
| 187818 | PEREZ ACERO JUAN PABLO | BACHILLERATO DE LA U.A.A |
| 174400 | PINEDA PEDRAZA RAUL IVAN | BACHILLERATO DE LA U.A.A |
| 237510 | PLASCENCIA RUIZ CHRISTIAN SAUL | CONALEP PLANTEL AGUASCALIENTES I (BIVALENTE |

| | | |
|---|---|---|
| 234800 | QUINTANILLA GONZALEZ JUAN PABLO | C.B.T.I.S. NO. 168 |
| 189775 | QUINTERO MARTINI RODOLFO JAVIER | BACHILLERATO DE LA U.A.A |
| 240687 | ROMO HUERTA JOSE ALBERTO | INSTITUTO LATINOAMERICANO DE SAN LUIS, A.C |
| 221878 | SANCHEZ CASTAÑEDA MIGUEL ANGEL | C.E.T.I.S. NO. 155 |
| 234961 | SANDOVAL GARCIA RUBEN OMAR | C.B.T.I.S. NO. 168 |
| 238858 | SOTO DAVILA VICTOR ADRIAN | C.B.T.I.S. NO. 39 |
| 188885 | VARELA DAVILA RICARDO | BACHILLERATO DE LA U.A.A |
| 219055 | VELASCO GONZALEZ MYRIAM ALEJANDRA | C.E.T.I.S. NO. 80 |
| 242348 | VIRAMONTES JUAREZ CHRISTIAN ANTONIO | C.B.T.I.S. NO. 39 |
| 183668 | ZUÑIGA RIVERA MIGUEL OMAR | INSTITUTO AGUASCALIENTES A.C |

7 CBTIS 168

| # Alumno | CBTIS 168, programacion |
|---|---|
| 10/56 | Grupo de control |
| 7/50 | Grupo experimental |

## Anexo 6: Tablas de distribución de frecuencias

Tablas de distribución de frecuencias de exámenes parciales, grupo experimental y de control.

Distribución de frecuencias resultados primer parcial, grupo experimental.

**PrimerParcialExp**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | 43.40 | 1 | 2.1 | 2.4 | 2.4 |
| | 45.00 | 1 | 2.1 | 2.4 | 4.8 |
| | 53.50 | 1 | 2.1 | 2.4 | 7.1 |
| | 56.50 | 2 | 4.3 | 4.8 | 11.9 |
| | 58.00 | 1 | 2.1 | 2.4 | 14.3 |
| | 64.00 | 1 | 2.1 | 2.4 | 16.7 |
| | 65.50 | 1 | 2.1 | 2.4 | 19.0 |
| | 70.00 | 1 | 2.1 | 2.4 | 21.4 |
| | 71.50 | 1 | 2.1 | 2.4 | 23.8 |
| | 73.40 | 1 | 2.1 | 2.4 | 26.2 |
| | 74.60 | 1 | 2.1 | 2.4 | 28.6 |
| | 76.00 | 1 | 2.1 | 2.4 | 31.0 |
| | 76.50 | 2 | 4.3 | 4.8 | 35.7 |
| | 76.60 | 1 | 2.1 | 2.4 | 38.1 |
| | 77.00 | 1 | 2.1 | 2.4 | 40.5 |
| | 79.50 | 1 | 2.1 | 2.4 | 42.9 |
| | 80.50 | 1 | 2.1 | 2.4 | 45.2 |
| | 81.00 | 1 | 2.1 | 2.4 | 47.6 |
| | 81.50 | 1 | 2.1 | 2.4 | 50.0 |
| | 81.60 | 1 | 2.1 | 2.4 | 52.4 |
| | 82.00 | 2 | 4.3 | 4.8 | 57.1 |
| | 83.00 | 2 | 4.3 | 4.8 | 61.9 |
| | 84.50 | 1 | 2.1 | 2.4 | 64.3 |
| | 85.00 | 2 | 4.3 | 4.8 | 69.0 |
| | 86.60 | 1 | 2.1 | 2.4 | 71.4 |
| | 87.00 | 2 | 4.3 | 4.8 | 76.2 |
| | 87.50 | 1 | 2.1 | 2.4 | 78.6 |
| | 90.00 | 1 | 2.1 | 2.4 | 81.0 |
| | 90.50 | 2 | 4.3 | 4.8 | 85.7 |
| | 91.50 | 1 | 2.1 | 2.4 | 88.1 |
| | 93.00 | 3 | 6.4 | 7.1 | 95.2 |
| | 93.50 | 1 | 2.1 | 2.4 | 97.6 |
| | 97.00 | 1 | 2.1 | 2.4 | 100.0 |
| | Total | 42 | 89.4 | 100.0 | |
| Perdidos | Sistema | 5 | 10.6 | | |
| Total | | 47 | 100.0 | | |

*Tabla 14: Distribución de frecuencias resultados primer parcial, grupo experimental*

Distribución de frecuencias resultados primer parcial, grupo de control

**PrimerParcialCtrl**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | 23.50 | 1 | 2.1 | 2.1 | 2.1 |
| | 40.00 | 1 | 2.1 | 2.1 | 4.3 |
| | 48.50 | 1 | 2.1 | 2.1 | 6.4 |
| | 49.50 | 1 | 2.1 | 2.1 | 8.5 |
| | 52.50 | 1 | 2.1 | 2.1 | 10.6 |
| | 68.50 | 1 | 2.1 | 2.1 | 12.8 |
| | 69.00 | 1 | 2.1 | 2.1 | 14.9 |
| | 69.50 | 1 | 2.1 | 2.1 | 17.0 |
| | 71.00 | 1 | 2.1 | 2.1 | 19.1 |
| | 72.50 | 1 | 2.1 | 2.1 | 21.3 |
| | 76.50 | 2 | 4.3 | 4.3 | 25.5 |
| | 77.50 | 1 | 2.1 | 2.1 | 27.7 |
| | 78.00 | 1 | 2.1 | 2.1 | 29.8 |
| | 79.50 | 2 | 4.3 | 4.3 | 34.0 |
| | 81.50 | 2 | 4.3 | 4.3 | 38.3 |
| | 83.50 | 1 | 2.1 | 2.1 | 40.4 |
| | 84.00 | 1 | 2.1 | 2.1 | 42.6 |
| | 86.50 | 1 | 2.1 | 2.1 | 44.7 |
| | 89.00 | 1 | 2.1 | 2.1 | 46.8 |
| | 89.50 | 1 | 2.1 | 2.1 | 48.9 |
| | 91.00 | 1 | 2.1 | 2.1 | 51.1 |
| | 91.50 | 1 | 2.1 | 2.1 | 53.2 |
| | 92.00 | 1 | 2.1 | 2.1 | 55.3 |
| | 94.00 | 2 | 4.3 | 4.3 | 59.6 |
| | 95.50 | 2 | 4.3 | 4.3 | 63.8 |
| | 97.00 | 1 | 2.1 | 2.1 | 66.0 |
| | 98.00 | 1 | 2.1 | 2.1 | 68.1 |
| | 99.00 | 2 | 4.3 | 4.3 | 72.3 |
| | 99.50 | 1 | 2.1 | 2.1 | 74.5 |
| | 100.00 | 12 | 25.5 | 25.5 | 100.0 |
| | Total | 47 | 100.0 | 100.0 | |

*Tabla 15: Distribución de frecuencias resultados primer parcial, grupo de control*

Distribución de frecuencias resultados segundo parcial, grupo experimental

**SegundoParcialExp**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | 21.60 | 1 | 2.1 | 2.4 | 2.4 |
| | 24.30 | 1 | 2.1 | 2.4 | 4.8 |
| | 37.70 | 1 | 2.1 | 2.4 | 7.1 |
| | 44.00 | 1 | 2.1 | 2.4 | 9.5 |
| | 51.30 | 1 | 2.1 | 2.4 | 11.9 |
| | 61.50 | 1 | 2.1 | 2.4 | 14.3 |
| | 65.00 | 1 | 2.1 | 2.4 | 16.7 |
| | 67.75 | 1 | 2.1 | 2.4 | 19.0 |
| | 68.00 | 1 | 2.1 | 2.4 | 21.4 |
| | 68.50 | 1 | 2.1 | 2.4 | 23.8 |
| | 68.75 | 1 | 2.1 | 2.4 | 26.2 |
| | 68.95 | 1 | 2.1 | 2.4 | 28.6 |
| | 70.20 | 1 | 2.1 | 2.4 | 31.0 |
| | 70.75 | 1 | 2.1 | 2.4 | 33.3 |
| | 72.00 | 1 | 2.1 | 2.4 | 35.7 |
| | 72.20 | 1 | 2.1 | 2.4 | 38.1 |
| | 72.60 | 1 | 2.1 | 2.4 | 40.5 |
| | 73.50 | 1 | 2.1 | 2.4 | 42.9 |
| | 73.85 | 1 | 2.1 | 2.4 | 45.2 |
| | 74.30 | 1 | 2.1 | 2.4 | 47.6 |
| | 75.00 | 1 | 2.1 | 2.4 | 50.0 |
| | 76.75 | 1 | 2.1 | 2.4 | 52.4 |
| | 77.80 | 1 | 2.1 | 2.4 | 54.8 |
| | 78.20 | 1 | 2.1 | 2.4 | 57.1 |
| | 80.75 | 1 | 2.1 | 2.4 | 59.5 |
| | 81.95 | 1 | 2.1 | 2.4 | 61.9 |
| | 84.25 | 1 | 2.1 | 2.4 | 64.3 |
| | 87.50 | 1 | 2.1 | 2.4 | 66.7 |
| | 88.00 | 1 | 2.1 | 2.4 | 69.0 |
| | 90.00 | 2 | 4.3 | 4.8 | 73.8 |
| | 91.75 | 1 | 2.1 | 2.4 | 76.2 |
| | 92.50 | 1 | 2.1 | 2.4 | 78.6 |
| | 92.70 | 1 | 2.1 | 2.4 | 81.0 |
| | 93.00 | 1 | 2.1 | 2.4 | 83.3 |
| | 94.25 | 1 | 2.1 | 2.4 | 85.7 |
| | 95.25 | 1 | 2.1 | 2.4 | 88.1 |
| | 96.00 | 1 | 2.1 | 2.4 | 90.5 |
| | 96.50 | 1 | 2.1 | 2.4 | 92.9 |
| | 97.25 | 1 | 2.1 | 2.4 | 95.2 |
| | 98.60 | 1 | 2.1 | 2.4 | 97.6 |
| | 99.25 | 1 | 2.1 | 2.4 | 100.0 |
| | Total | 42 | 89.4 | 100.0 | |
| Perdidos | Sistema | 5 | 10.6 | | |
| Total | | 47 | 100.0 | | |

*Tabla 16: Distribución de frecuencias resultados segundo parcial, grupo experimental*

Distribución de frecuencias resultados segundo parcial, grupo de control

**SegundoParcialCtrl**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | 10.50 | 1 | 2.1 | 2.1 | 2.1 |
| | 17.25 | 1 | 2.1 | 2.1 | 4.3 |
| | 32.75 | 1 | 2.1 | 2.1 | 6.4 |
| | 33.75 | 1 | 2.1 | 2.1 | 8.5 |
| | 35.75 | 1 | 2.1 | 2.1 | 10.6 |
| | 36.00 | 1 | 2.1 | 2.1 | 12.8 |
| | 39.75 | 1 | 2.1 | 2.1 | 14.9 |
| | 42.50 | 1 | 2.1 | 2.1 | 17.0 |
| | 46.00 | 1 | 2.1 | 2.1 | 19.1 |
| | 48.00 | 1 | 2.1 | 2.1 | 21.3 |
| | 48.25 | 1 | 2.1 | 2.1 | 23.4 |
| | 49.25 | 1 | 2.1 | 2.1 | 25.5 |
| | 51.25 | 1 | 2.1 | 2.1 | 27.7 |
| | 54.50 | 1 | 2.1 | 2.1 | 29.8 |
| | 55.00 | 1 | 2.1 | 2.1 | 31.9 |
| | 55.25 | 1 | 2.1 | 2.1 | 34.0 |
| | 55.80 | 1 | 2.1 | 2.1 | 36.2 |
| | 58.50 | 1 | 2.1 | 2.1 | 38.3 |
| | 62.00 | 1 | 2.1 | 2.1 | 40.4 |
| | 66.00 | 1 | 2.1 | 2.1 | 42.6 |
| | 67.25 | 1 | 2.1 | 2.1 | 44.7 |
| | 69.50 | 1 | 2.1 | 2.1 | 46.8 |
| | 70.00 | 1 | 2.1 | 2.1 | 48.9 |
| | 75.00 | 1 | 2.1 | 2.1 | 51.1 |
| | 75.25 | 1 | 2.1 | 2.1 | 53.2 |
| | 77.50 | 1 | 2.1 | 2.1 | 55.3 |
| | 79.50 | 1 | 2.1 | 2.1 | 57.4 |
| | 80.25 | 1 | 2.1 | 2.1 | 59.6 |
| | 81.00 | 1 | 2.1 | 2.1 | 61.7 |
| | 81.75 | 2 | 4.3 | 4.3 | 66.0 |
| | 82.25 | 1 | 2.1 | 2.1 | 68.1 |
| | 82.50 | 1 | 2.1 | 2.1 | 70.2 |
| | 85.50 | 1 | 2.1 | 2.1 | 72.3 |
| | 85.75 | 1 | 2.1 | 2.1 | 74.5 |
| | 86.00 | 1 | 2.1 | 2.1 | 76.6 |
| | 88.50 | 1 | 2.1 | 2.1 | 78.7 |
| | 90.75 | 1 | 2.1 | 2.1 | 80.9 |
| | 92.25 | 1 | 2.1 | 2.1 | 83.0 |
| | 92.50 | 1 | 2.1 | 2.1 | 85.1 |
| | 92.55 | 1 | 2.1 | 2.1 | 87.2 |
| | 93.25 | 1 | 2.1 | 2.1 | 89.4 |
| | 93.50 | 3 | 6.4 | 6.4 | 95.7 |
| | 95.50 | 1 | 2.1 | 2.1 | 97.9 |
| | 96.00 | 1 | 2.1 | 2.1 | 100.0 |
| | Total | 47 | 100.0 | 100.0 | |

*Tabla 17: Distribución de frecuencias resultados segundo parcial, grupo de control*

Distribución de frecuencias resultados tercer parcial, grupo experimental

**GpoExp**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | 10.50 | 1 | 2.1 | 2.4 | 2.4 |
| | 18.50 | 1 | 2.1 | 2.4 | 4.8 |
| | 22.50 | 1 | 2.1 | 2.4 | 7.1 |
| | 35.75 | 1 | 2.1 | 2.4 | 9.5 |
| | 40.50 | 1 | 2.1 | 2.4 | 11.9 |
| | 51.75 | 2 | 4.3 | 4.8 | 16.7 |
| | 63.05 | 1 | 2.1 | 2.4 | 19.0 |
| | 65.10 | 1 | 2.1 | 2.4 | 21.4 |
| | 66.30 | 1 | 2.1 | 2.4 | 23.8 |
| | 66.55 | 1 | 2.1 | 2.4 | 26.2 |
| | 67.00 | 1 | 2.1 | 2.4 | 28.6 |
| | 68.00 | 1 | 2.1 | 2.4 | 31.0 |
| | 68.80 | 1 | 2.1 | 2.4 | 33.3 |
| | 69.05 | 1 | 2.1 | 2.4 | 35.7 |
| | 69.10 | 1 | 2.1 | 2.4 | 38.1 |
| | 69.75 | 1 | 2.1 | 2.4 | 40.5 |
| | 71.75 | 1 | 2.1 | 2.4 | 42.9 |
| | 73.55 | 1 | 2.1 | 2.4 | 45.2 |
| | 75.05 | 1 | 2.1 | 2.4 | 47.6 |
| | 75.25 | 1 | 2.1 | 2.4 | 50.0 |
| | 75.50 | 1 | 2.1 | 2.4 | 52.4 |
| | 76.05 | 1 | 2.1 | 2.4 | 54.8 |
| | 76.60 | 1 | 2.1 | 2.4 | 57.1 |
| | 76.75 | 1 | 2.1 | 2.4 | 59.5 |
| | 77.00 | 1 | 2.1 | 2.4 | 61.9 |
| | 81.25 | 1 | 2.1 | 2.4 | 64.3 |
| | 82.50 | 1 | 2.1 | 2.4 | 66.7 |
| | 83.75 | 1 | 2.1 | 2.4 | 69.0 |
| | 85.00 | 2 | 4.3 | 4.8 | 73.8 |
| | 85.50 | 1 | 2.1 | 2.4 | 76.2 |
| | 87.50 | 1 | 2.1 | 2.4 | 78.6 |
| | 88.00 | 1 | 2.1 | 2.4 | 81.0 |
| | 89.00 | 1 | 2.1 | 2.4 | 83.3 |
| | 90.00 | 1 | 2.1 | 2.4 | 85.7 |
| | 92.00 | 1 | 2.1 | 2.4 | 88.1 |
| | 95.50 | 1 | 2.1 | 2.4 | 90.5 |
| | 98.25 | 1 | 2.1 | 2.4 | 92.9 |
| | 100.00 | 3 | 6.4 | 7.1 | 100.0 |
| | Total | 42 | 89.4 | 100.0 | |
| Perdidos | Sistema | 5 | 10.6 | | |
| Total | | 47 | 100.0 | | |

*Tabla 18: Distribución de frecuencias resultados tercer parcial, grupo experimental*

Distribución de frecuencias resultados tercer parcial, grupo de control

**GpoCtrl**

| | | Frecuencia | Porcentaje | Porcentaje válido | Porcentaje acumulado |
|---|---|---|---|---|---|
| Válidos | .00 | 4 | 8.5 | 8.5 | 8.5 |
| | 1.00 | 1 | 2.1 | 2.1 | 10.6 |
| | 5.05 | 1 | 2.1 | 2.1 | 12.8 |
| | 9.60 | 1 | 2.1 | 2.1 | 14.9 |
| | 10.00 | 1 | 2.1 | 2.1 | 17.0 |
| | 11.00 | 1 | 2.1 | 2.1 | 19.1 |
| | 13.00 | 1 | 2.1 | 2.1 | 21.3 |
| | 13.50 | 1 | 2.1 | 2.1 | 23.4 |
| | 15.50 | 1 | 2.1 | 2.1 | 25.5 |
| | 15.75 | 1 | 2.1 | 2.1 | 27.7 |
| | 28.85 | 1 | 2.1 | 2.1 | 29.8 |
| | 30.00 | 1 | 2.1 | 2.1 | 31.9 |
| | 41.75 | 1 | 2.1 | 2.1 | 34.0 |
| | 43.80 | 1 | 2.1 | 2.1 | 36.2 |
| | 44.35 | 1 | 2.1 | 2.1 | 38.3 |
| | 45.95 | 1 | 2.1 | 2.1 | 40.4 |
| | 47.25 | 1 | 2.1 | 2.1 | 42.6 |
| | 48.45 | 1 | 2.1 | 2.1 | 44.7 |
| | 49.90 | 1 | 2.1 | 2.1 | 46.8 |
| | 50.50 | 1 | 2.1 | 2.1 | 48.9 |
| | 51.95 | 1 | 2.1 | 2.1 | 51.1 |
| | 61.50 | 1 | 2.1 | 2.1 | 53.2 |
| | 62.75 | 1 | 2.1 | 2.1 | 55.3 |
| | 65.35 | 1 | 2.1 | 2.1 | 57.4 |
| | 71.10 | 1 | 2.1 | 2.1 | 59.6 |
| | 71.25 | 1 | 2.1 | 2.1 | 61.7 |
| | 73.25 | 1 | 2.1 | 2.1 | 63.8 |
| | 77.00 | 1 | 2.1 | 2.1 | 66.0 |
| | 77.10 | 1 | 2.1 | 2.1 | 68.1 |
| | 82.00 | 1 | 2.1 | 2.1 | 70.2 |
| | 82.75 | 1 | 2.1 | 2.1 | 72.3 |
| | 83.35 | 1 | 2.1 | 2.1 | 74.5 |
| | 83.40 | 1 | 2.1 | 2.1 | 76.6 |
| | 84.75 | 1 | 2.1 | 2.1 | 78.7 |
| | 85.20 | 1 | 2.1 | 2.1 | 80.9 |
| | 87.30 | 1 | 2.1 | 2.1 | 83.0 |
| | 87.75 | 1 | 2.1 | 2.1 | 85.1 |
| | 88.50 | 1 | 2.1 | 2.1 | 87.2 |
| | 89.80 | 1 | 2.1 | 2.1 | 89.4 |
| | 94.50 | 1 | 2.1 | 2.1 | 91.5 |
| | 94.95 | 1 | 2.1 | 2.1 | 93.6 |
| | 95.80 | 1 | 2.1 | 2.1 | 95.7 |
| | 99.15 | 1 | 2.1 | 2.1 | 97.9 |
| | 99.25 | 1 | 2.1 | 2.1 | 100.0 |
| | Total | 47 | 100.0 | 100.0 | |

*Tabla 19: Distribución de frecuencias resultados tercer parcial, grupo de control*