TESIS DE MAESTRÍA

Ingeniería de Sistemas de Información

Título:

“Propuesta de protocolo de formación de pares experimentales de programadores”

Autor: Esp. Mauricio Roberto Dávila

Director de Tesis: Dr. Darío Rodriguez

Codirector de Tesis: Mg. Marisa Panizzi

Buenos Aires - 2018

RESUMEN

Dentro del área de Ingeniería de Software Experimental, es habitual enfrentarse a la necesidad de realizar experimentos con un conjunto reducido de personas las cuales aplican diferentes tratamientos a los objetos de estudio. Dicha situación requiere tener en cuenta las diferencias de aptitudes que pudieran existir entre los participantes del experimento para que estas no afecten la interpretación de los resultados. En este contexto, el presente trabajo de tesis tiene como objetivo elaborar un protocolo que permita formar pares experimentales homogéneos de programadores, con el propósito de asegurar que dos sujetos de iguales características sean indistinguibles en lo que refiere a sus aptitudes como programadores. El protocolo permitirá evaluar características de los lenguajes de programación o de las herramientas utilizadas para programar, sin que estas mediciones se vean afectadas por las habilidades de los programadores que las emplean. Se presenta un caso testigo que permite validar el protocolo, el cual tiene como propósito demostrar que los pares experimentales formados no presentan diferencias significativas, en lo que refiere a calidad y tiempo, en la codificación de una especificación. Por último se aplica el protocolo a un caso, en el cual se busca determinar si un lenguaje de programación incide o no en la productividad de sujetos que son considerados homogéneos en lo que refiere a sus conocimientos y habilidades.

Palabras clave: experimentación en ingeniería de software, protocolo, pares experimentales de programadores, productividad, programación.

ABSTRACT

In the field of Experimental Software Engineering, it is usual to face the need to perform experiments with a small group of people that apply different treatments to the objects of study. This situation requires taking into account the differences in aptitudes that may exist among the participants of the experiment so that, they do not affect the results interpretation. In this context, this thesis work aims to develop a protocol that allows to form homogeneous experimental pairs of programmers, in order to ensure that the characteristics of the subjects are identical with regard to their skills as programmers. The protocol will allow to evaluate the characteristics of the programming languages or the tools used to program, without these measurements being affected by the skills of the programmers who use them. A test case is presented to validate the protocol, whose purpose is to demonstrate that the experimental pairs do not present significant differences, according to quality and time, in the coding of a specification. Finally, the protocol is applied to a case, which seeks to determine if a programming language affects or not the productivity of subjects that are considered homogeneous in relation to their knowledge and skills.

Keywords: experimentation in software engineering, protocol, experimental pairs of programmers, productivity, programming.

DEDICATORIA

A la memoria del Dr. Ramón García Martínez, por su calidad humana, su paciencia, su gran sentido del humor y sus acertados consejos en pos de mi crecimiento profesional.

AGRADECIMIENTOS

A la Escuela de Posgrado de la Universidad Tecnológica Nacional - Facultad Regional Buenos Aires, por haberme abierto sus puertas como “alma máter” para realizar mis estudios de Maestría.

A la Universidad Tecnológica Nacional - Facultad Regional Avellaneda por permitir desarrollarme como docente.

A mis colegas por haber contribuido abierta y desinteresadamente en las pruebas y consultas realizadas.

A mis alumnos, por lo que me enseñan y por haberme permitido crecer como docente.

A Marisa Panizzi, por su inestimable ayuda, su paciencia, sus acertadas observaciones y sugerencias.

A Darío Rodríguez por sus valiosos aportes y la confianza depositada en este trabajo.

A mi esposa, por la bondadosa paciencia conque sobrellevo todas las horas que le robé para poder llevar adelante esta tesis.

A mis padres, por haber priorizado mi educación por sobre todas las cosas.

A mis amigos y compañeros de trabajo quienes siempre intentaron apoyarme para la consecución de este logro académico.

A todos los que me ayudaron a llegar aquí…

INDICE

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

NOMENCLATURA

ACM	Association for Computing Machinery
CDLP	Caracterización Dependiente del Lenguaje de Programación
CDLP1	Experiencia con el Lenguaje Elegido
CDLP2	Conocimiento Teórico
CDLP3	Comprensión de Código Fuente
CILP	Caracterización Independiente del Lenguaje de Programación
CILP1	Nivel de formación
CILP2	Experiencia
CILP3	Comprensión de una Especificación
CILP4	Comprensión de Pseudocódigo
CILP5	Capacidad Algorítmica
CONICET	Consejo Nacional de Investigaciones Científicas y Técnicas
IEEE	Institute of Electrical and Electronics Engineers
OMS	Organización Mundial de la Salud

1. INTRODUCCIÓN

En este capítulo se presenta el contexto de la tesis (sección 1.1), se establecen sus objetivos (sección 1.2), se describe la metodología empleada (sección 1.3), se presentan las publicaciones del tesista vinculadas a las investigaciones realizadas en el desarrollo de la tesis (sección 1.4) y se resume la estructura de la tesis (sección 1.5).

1.1. Contexto de la Tesis

La construcción de sistemas de información demanda grandes inversiones de tiempo y dinero, por lo cual se torna necesario identificar los factores clave que conducen a un mejor y más productivo desarrollo de estos. Según [Bhattacharya, 2011] uno de estos factores clave, es la correcta elección del o los lenguajes de programación que intervendrán en el proyecto. En la actualidad existe una gran diversidad de lenguajes de programación y a menudo esto dificulta la tarea de seleccionar el lenguaje que mejor se adapta a las necesidades del desarrollo. La decisión de que lenguaje de programación utilizar para desarrollar una solución implica múltiples factores de análisis, muchos de los cuales están sujetos al código fuente producido y al tiempo empleado para poder producirlo. Tradicionalmente, la productividad ha sido definida como la relación de la salida producida por unidad de entrada [Jefferys et al., 1954]; donde los insumos son los recursos necesarios para producir los productos obtenidos. Según el diccionario de la Real Academia Española [RAE, 2016], la definición económica de la productividad refiere a un concepto que describe la relación entre lo producido y los medios empleados para hacerlo. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) ha publicado en 1992 un estándar para las métricas de productividad de software [IEEE Std 1045, 1992] allí define la productividad como la relación entre una salida primitiva (líneas de código fuente, puntos de función o documentos) y su correspondiente entrada primitiva (esfuerzo , por ejemplo, horas hombre). El concepto de entradas y salidas aparecen tanto en las definiciones de productividad referidas a la Ingeniería del Software como en las definiciones tradicionales. Es aceptado que factores tales como: las limitaciones de tiempo, los requisitos de fiabilidad, los lenguajes de programación, el tamaño del equipo de desarrollo, la volatilidad de los requisitos, la habilidad del personal con las herramientas, la disponibilidad de personal, la participación del cliente, y la duración del proyecto influyen en la productividad [Foulds et al., 2007; Maxwell y Forselius, 2000]. A pesar de su aceptación, la influencia, positiva o negativa, de algunos de estos factores en la productividad no es clara.

Este estudio se centrara en analizar uno de los factores que afecta la productividad, el referido al lenguaje de programación empleado. [Abelson et al., 1996] expresa que un lenguaje de programación sirve como un marco dentro del cual un programador organiza sus ideas acerca de los procesos y es el lenguaje quien proporciona los medios que permiten articular soluciones simples en pos de resolver problemas complejos. Se considera que existe la necesidad de realizar estudios comparativos sobre la influencia en la productividad de los lenguajes de programación utilizados para desarrollar artefactos de software.

1.2. Objetivos de la Tesis

Los objetivos de esta tesis se presentan en un objetivo general a alcanzar (sección 1.2.1) y un conjunto de objetivos específicos (sección 1.2.2) que definen los pasos a seguir para lograr el objetivo general.

1.2.1. Objetivo General

El objetivo general de este trabajo es definir un protocolo que permita formar pares experimentales homogéneos de programadores con el propósito de asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. Con el fin de poder evaluar características de los lenguajes de programación sin que estas mediciones se vean afectadas por el desempeño de los programadores que los utilizan.

1.2.2. Objetivos Específicos

Se detallan a continuación los objetivos específicos que permiten, en conjunto, establecer los pasos a seguir para lograr cumplir con el objetivo general:

- Realizar un análisis documental en lo que refiere a los beneficios que la formación de pares experimentales aporta en el área de ingeniería de software.
- Realizar un análisis documental en lo que refiere a los aspectos éticos en relación a las investigaciones en el campo de la ingeniería de software con el fin de tener estos en cuenta en la investigación.
- Llevar adelante la construcción de los instrumentos que permitan caracterizar a los programadores.
- Definir el procedimiento que permite formar los pares experimentales a partir de los datos obtenidos con los instrumentos de caracterización.

- Verificar mediante un caso la validez del protocolo de formación de pares experimentales de programadores.
- Aplicar el protocolo de formación de pares experimentales de programadores a un caso.

1.3. Metodología Empleada

Para construir el conocimiento de la presente investigación, se seguirá un enfoque de investigación clásico [Riveros y Rosas, 1985] con énfasis en la producción de tecnologías [Sábato y Mackenzie, 1982] ; identificando los métodos y materiales necesarios para desarrollar el proyecto.

1.3.1. Métodos

A continuación se definen los métodos que se llevarán a cabo en el presente trabajo.

1.3.1.1. Revisiones sistemáticas

Las revisiones sistemáticas [Argimón, 2004] de artículos científicos siguen un método explícito para resumir la información sobre determinado tema o problema; se diferencia de las revisiones narrativas en que provienen de una pregunta estructurada y de un protocolo previamente realizado.

1.3.1.2. Prototipado evolutivo experimental, método de la ingeniería

El prototipado evolutivo experimental [Basili, 1993] consiste en desarrollar una solución inicial para un determinado problema, generando su refinamiento de manera evolutiva por prueba de aplicación de dicha solución a casos de estudio (problemáticas) de complejidad creciente. El proceso de refinamiento concluye al estabilizarse el prototipo en evolución. Para la construcción de las herramientas de caracterización se tuvo en cuenta el procedimiento general de construcción de un instrumento de medición propuesto por [Sampieri et al., 2010], adaptando dicho procedimiento a la necesidad de este trabajo.

1.3.1.3. Test de Wilcoxon

El método de Wilcoxon permite analizar la similitud entre un conjunto de datos muestrales apareados donde cada elemento de la población posee un valor experimental que se desea comprobar y un valor de referencia o de control [Guardía-Olmos et al., 2007]. Como resultado de la prueba, es posible aceptar o refutar una hipótesis nula (H0) la cual indica que la población de diferencias entre los valores experimentales y los valores de control tienen una mediana de cero [Triola, 2013]. Se prevé utilizar el Método de Wilcoxon [1945] en esta tesis por cuanto se busca

contrastar la distribución de un conjunto de variables que caracterizan un grupo tipo de programadores, a efecto de determinar si las mismas presentan diferencias significativas entre el grupo que utiliza el lenguaje A y el que utiliza el lenguaje B.

1.3.1.4. Materiales

Conjunto de ejercicios de programación de complejidad creciente susceptibles de ser resueltos utilizando los lenguajes de programación considerados.

1.3.2. Metodología

Para alcanzar los Objetivos trazados se propone:

i. Realizar una investigación documental exploratoria sobre las métricas utilizadas en relación con los lenguajes de programación.

ii. Realizar una investigación documental exploratoria sobre los aspectos relevantes a la hora de categorizar a un programador, identificar caso de estudio.

iii. Desarrollar mediante el método de prototipado evolutivo un protocolo que permita valorar elementos a ser utilizados para categorizar a los programadores.

iv. A partir de los problemas desarrollados y los grupos de trabajo identificados, desarrollar la experiencia de resolución de los problemas tomando las mediciones correspondientes.

v. Evaluar e interpretar resultados.

vi. Redactar la memoria de la Tesis.

1.4. Producción Científica Derivada de Resultados Parciales de la Tesis

Durante el desarrollo de esta tesis se han comunicado resultados parciales a través de las siguientes publicaciones:

Artículos en Revistas con Referato:

- Dávila, M. 2016. Investigación en Progreso: Estudio Comparativo de la Incidencia de los Lenguajes de Programación en la Productividad Informática. Revista Latinoamericana de Ingeniería de Software, 4(6), Pág. 255-258. ISSN 2314-2642.

Comunicaciones a Congresos:

- Dávila, M., Panizi, M. y Rodriguez, D. 2017. Propuesta de Protocolo de Formación de Pares Experimentales de Programadores. Libro de Actas del XXIII Congreso Argentino de Ciencias de la Computación. Pág. 782-791. ISBN 978-950-34-1539-9. 9 al 13 de Octubre. Universidad Nacional de La Plata. Argentina.

- Dávila, M., Panizi, M. y Rodriguez, D. 2017. Aproximación a un Protocolo de Formación de Pares Experimentales de Programadores. Memorias del 5to Congreso Nacional de Ingeniería Informática / Sistemas de Información. ISSN 2347-0372. 2 y 3 de Noviembre. Facultad Regional Santa Fe. Universidad Tecnológica Nacional. Argentina.

- Dávila, M., Panizi, M. y Rodriguez, D. 2017. Proposal for the Formation of Experimental Pair Programmers. In Argentine Congress of Computer Science (pp. 135-144). Springer, Cham.

1.5. Estructura General del Trabajo

En el capítulo 1 “Introducción” se plantea el contexto de la tesis, se establece el objetivo general del trabajo y su objetivos particulares, se presentan las metodologías a emplear, se enumeran las publicaciones del tesista vinculadas a las investigaciones realizadas en el desarrollo de la tesis y se resume la estructura de la tesis.

En el capítulo 2 “Estado de la Cuestión” se presenta la descripción de los principales conceptos relacionados a este trabajo de tesis. Comenzando por cuantificar la cantidad de lenguajes vigentes en la actualidad, se identifican diversas métricas utilizadas en la ingeniería de software y se analiza el propósito de dichas métricas. Posteriormente, se plantea la importancia de considerar el factor humano a la hora de evaluar un lenguaje de programación y la problemática de no contar con mecanismos que lo tengan en cuenta a la hora de evaluar lenguajes de programación. Finalmente, se plantea la importancia de la experimentación en el campo de la ingeniería de software.

En el capítulo 3 “Planteamiento del Problema” se identifican los problemas de investigación a ser resueltos en este trabajo de tesis con un sumario de investigación. Estos surgen por las dificultades relacionados con la evaluación de las características de los lenguajes de programación y la búsqueda

de que dichas características no se vean afectadas por el desempeño de los programadores que los utilizan.

En el capítulo 4 “Aspectos Éticos de la Investigación” se analizan los aspectos éticos que deben de considerarse en una investigación en el área de ingeniería de software en la cual intervienen personas a fin de poder ser aplicados a la presente investigación.

En el capítulo 5 “Solución” se propone un protocolo que permite formar pares experimentales homogéneos de programadores, cada propuesta incluye un proceso, una lista de características a evaluar y métodos para realizar los cálculos necesarios.

En el capítulo 6 “Caso de Aplicación” se realiza un experimento con el fin de aplicar el protocolo de formación de pares experimentales de programadores a un caso real en el cual se busca determinar si un lenguaje de programación es más productivo que otro.

En el capítulo 7 “Conclusiones y Futuras Líneas de Investigación” se presentan las aportaciones de esta tesis y se destacan las futuras líneas de investigación, las cuales se consideran de interés en base al problema abierto que se aborda en este trabajo de tesis.

Finalmente, en el capítulo 8 “Referencias” se listan todas las publicaciones consultadas para el desarrollo de esta tesis.

En el “Anexo A” se incluye el formulario de consentimiento libre e informado utilizado por los participantes.

En el “Anexo B” se describe la prueba de rangos con signo de Wilcoxon [1945] la cual es aplicada en el capítulo 6 para llevar a cabo la validación de los modelos propuestos.

En el “Anexo C” se incluye el formulario con la descripción de la tarea ser resuelta como parte del “Caso de Aplicación” del capítulo 6.

En el “Anexo D” se incluye el cuestionario que es utilizado para realizar la Caracterización Independiente del Lenguaje de Programación.

En el “Anexo E” se incluye el cuestionario que es utilizado para realizar la Caracterización Dependiente del Lenguaje de Programación C.

En el “Anexo F” se incluye el cuestionario que es utilizado para realizar la Caracterización Dependiente del Lenguaje de Programación C#.

En el “Anexo G” se describe el algoritmo de selección de parejas experimentales el cual es aplicado en el capítulo 5 y 6 para llevar a cabo la validación de los modelos propuestos.

2. ESTADO DE LA CUESTIÓN

La problemática que enfrenta esta tesis es la de elaborar mecanismos que permitan comparar características de los lenguajes de programación sin que estas mediciones se vean afectadas por el desempeño de los programadores que los utilizan. En la (sección 2.1) se comienza por cuantificar la cantidad de lenguajes vigentes en la actualidad, en la (sección 2.2) se analiza el propósito de las métricas de software existentes, en la (sección 2.3) se destaca la importancia de considerar el factor humano a la hora de evaluar un lenguaje de programación y la problemática de no contar con mecanismos que lo tengan en cuenta a la hora de evaluar lenguajes de programación, y por último en la (sección 2.4) se acentúa la importancia de la experimentación en el campo de la ingeniería de software.

2.1. Lenguajes de Programación

La creación de nuevos lenguajes de programación y la evolución de los existente es incesante, tomando como punto de partida la década del cuarenta donde surgen las primeras computadoras modernas hasta la actualidad, se han creado una gran cantidad de leguajes de programación de alto nivel. Kinnersley [2016] realiza un trabajo de recopilación que reúne información de más de dos mil quinientos lenguajes. Muchos lenguajes de programación han quedado en desuso (ALGOL 60, CPL, FACT, entre otros) [Sammet, 1972; Lévénez, 2015], otros de los surgidos muchos años atrás han evolucionado e incorporado nuevas características [Cobol, 2017; Fortran, 2017], también han surgido nuevos lenguajes que han logrado un lugar en el mercado [Erlang, 2017; Java, 2017; Python, 2017] y continúan surgiendo otros que buscan posicionarse [Hack, 2017; Swift, 2017]. Para determinar qué lenguajes se encuentran en uso, se puede tomar como parámetro que, al día de hoy, en el repositorio GitHub existe código escrito en casi trecientos lenguajes de programación distintos [Zapponi, 2016] y que poco más de doscientos cuarenta lenguajes son tomados en cuenta para elaborar el índice TIOBE [2016] el cual mide el volumen de información disponible en la web sobre cada lenguaje. Michael Scott [2009] entiende que la existencia de tantos lenguajes de programación puede responder a tres motivos:

- Evolución: La informática es una disciplina joven; en constante búsqueda de mejores formas de hacer las cosas.
- Propósitos Especiales: Muchos lenguajes fueron diseñados para un dominio del problema específico.
- Preferencia personal: A diferentes personas les gustan cosas diferentes.

A pesar de existir una gran cantidad de lenguajes no todos son ampliamente utilizados. La IEEE elabora un índice denominado “The Top Programming Languages 2015” [Cass, 2015], formado por una lista con los cuarenta y ocho lenguajes más populares. En su libro Scott [2009] considera que los siguientes factores influyen en la popularidad de un lenguaje de programación:

- Potencia expresiva: tienen un gran impacto en la capacidad del programador para escribir código claro, conciso y fácil de mantener.
- Facilidad de uso para el usuario principiante: en muchos casos el éxito se debe a una muy baja "curva de aprendizaje".
- Facilidad de implementación: Algunos lenguajes son exitosos ya que se pueden implementar con facilidad en diversos dispositivos.
- Normalización: Casi todos los lenguajes ampliamente utilizados tienen una norma internacional oficial o (en el caso de varios lenguajes de scripting) una sola aplicación canónica.
- Código abierto: La mayoría de los lenguajes de programación de hoy tienen al menos un compilador/ intérprete de código abierto.
- Compiladores/Intérpretes: muchos lenguajes tienen éxito, en parte porque tienen compiladores y herramientas de apoyo que hacen un buen trabajo.
- Economía, patrocinio, y la inercia: Por último, hay otros factores además de los méritos técnicos que influyen en gran medida el éxito. El respaldo de un poderoso patrocinador es uno.

En este escenario, con una gran diversidad de lenguajes de programación, algunos más populares que otros, se dificulta la tarea de seleccionar el lenguaje adecuado para desarrollar una solución determinada.

2.2. Métricas de Software

Algunos estudios [Prechelt, 2000; Fulgham y Gouy, 2009; Nanz y Furia, 2015] tratan de determinar cuál es el lenguaje adecuado para resolver un problema mediante la comparación de características de desempeño como ser: uso de CPU, uso de Memoria o Tiempos de ejecución. Pero dejan de lado que características como las antes mencionadas son sólo relevantes en entornos en los cuales los recursos disponibles son limitados y es posible llegar conclusiones del estilo: “El lenguaje A demora menos en realizar una tarea X que el lenguaje B”. En entornos de “Cloud Computing” [Amazon, 2017; Google Cloud, 2017] los recursos son prácticamente ilimitados, la computación en la nube es un paradigma bien consolidado para el aprovisionamiento de servicios bajo demanda, a

este concepto se lo conoce con el nombre de elasticidad y refiere a la capacidad de agregar y quitar recursos "sobre la marcha " para manejar la variación de la carga [Coutinho at el, 2015]. Otras características a ser tenidas en cuenta a la hora de comparar lenguajes son las constructivas [Harrison et al., 1996; Abelson et al., 1996; Scott, 2000; Watt, 2004; Sebesta, 2012], aquellas que nos indican como es el lenguaje y como éste realiza las cosas:

- Ejecución de forma nativa o en una máquina virtual
- Manejo de la memoria
- Manejo de errores
- Plataformas que lo soporta
- Ecosistema de las bibliotecas
- Paradigma

Las métricas de software proporcionan información relevante a tiempo que contribuye a gestionar de forma más efectiva un proyecto, y mejorar la calidad de los procesos y productos de software, estas contemplan varias clasificaciones que apuntan a diferentes aspectos del proceso y del producto de software [Pressman, 2015].

- Métricas del producto: son métricas que evalúan la calidad de los productos entregables, permitiendo tener un conocimiento detallado del diseño y la construcción del producto software. En estas métricas se tienen en cuenta atributos como: tamaño, calidad, complejidad, esfuerzo, volatilidad, entre otros.

- Métricas del proceso: son métricas aplicadas a fines estratégicos y propician indicadores que conducen a avances en el proceso y ambiente de desarrollo del software, a partir de información histórica de procesos similares. Se utilizan para evaluar si la eficiencia de un proceso ha mejorado en el largo plazo.

- Métricas del proyecto: son métricas de tipo tácticas y describen las características propias del proyecto y de su ejecución. A su vez, permiten evaluar la calidad de los productos obtenidos en cada etapa del desarrollo [McDermid, 1991]. Estas métricas tienen en cuenta atributos como duración real del proyecto, esfuerzo real [persona-mes] por proceso, subproceso y por proyecto, progreso del proyecto, tamaño del proyecto, costo total invertido, entre otros.

Existe un gran número de métricas para capturar atributos de los procesos y productos software, que tradicionalmente se han realizado confiando en la sabiduría de los expertos y esta situación ha conducido frecuentemente a cierto grado de imprecisión en las definiciones, propiedades y suposiciones de las métricas, haciendo que el uso de las métricas sea difícil, la interpretación

peligrosa y los resultados de muchos estudios de validación contradictorios [Genero, M. et al., 2014]. En su trabajo [Basso, 2014] enumera un conjunto de métricas existentes en la ingeniería de software, de acuerdo al aspecto y atributo del software que miden. La (Tabla 2.1) corresponde a las métricas de producto, la (Tabla 2.2) a las métricas de proyecto y la (Tabla 2.3) a las métricas de proceso.

Atributo	Métricas Existentes
Tamaño	Líneas de Código (medidas en miles - KLDC)
	Puntos de Función (PF)
	Páginas de Documentación
Complejidad	Complejidad ciclomática
	Nivel de acoplamiento de los módulos
	Nivel de modularidad (cohesión de módulos)
Calidad	Cantidad de defectos por KLDC
	Cantidad de errores encontrados por KLDC
	Cantidad de defectos/errores que encuentran los usuarios después de la entrega
	Tipo y origen de los defectos (requerimientos, análisis y diseño, construcción, integración y pruebas)
Mantenimiento	Cantidad de componentes
	Volatilidad de los componentes
	Complejidad de los componentes
	Cantidad de requerimientos nuevos, de cambios o mejoras
	Cantidad de requerimientos de corrección de defectos
	Tiempo promedio de corrección de errores o defectos
	Tiempo promedio de cambios
	Porcentaje del código corregido
Confiabilidad	Tiempo transcurrido entre fallas
	Tiempo esperado entre fallas
	Tiempo requerido para corregir una falla
	Nivel de severidad de la falla
Usabilidad	Facilidad de aprendizaje de uso
	Errores cometidos por los usuarios con el uso
	Tiempo requerido para realizar las tareas
Rendimiento	Tiempos de respuesta
	Utilización de recursos
	Tiempo de recuperación

Tabla 2.1. Métricas de producto existentes en Ingeniería de Software [Basso,2014].

Atributo	Métricas Existentes
Esfuerzo	Cantidad de horas trabajadas
	Cantidad de personas que trabajan en el proyecto
	Tiempo transcurrido
	Distribución del esfuerzo por fase
Costo	Costo del Desarrollo
	Costo del Soporte
	Costo de hs/persona
Productividad	Cantidad de software desarrollado por unidad de tiempo de trabajo
	Tamaño/Esfuerzo
	Ritmo de entrega del software por unidad de tiempo transcurrido.
Seguimiento	Cronograma real vs Cronograma estimado
	Porcentaje de tareas completadas
	Porcentaje de requerimientos implementados por unidad de tiempo
	Porcentaje de tiempo total dedicado a las pruebas
	Porcentaje de error en la estimación del tiempo
	Costo sobre el valor agregado
Estabilidad	Origen de los cambios en los requerimientos
	Cambios de los requerimientos en el desarrollo
	Cambios en los requerimientos en producción

Tabla 2.2. Métricas de proyecto existentes en Ingeniería de Software [Basso,2014].

Atributo	Métricas Existentes
Esfuerzo	Distribución del esfuerzo por fase del proceso
	Cantidad de personas requeridas
	Esfuerzo requerido para corregir un defecto
	Esfuerzo requerido para mejorar un defecto
Reusabilidad	Cantidad de componentes reutilizados
	Grado de reusabilidad de los componentes
Calidad	Cantidad de defectos sin corregir
	Costo de corrección de defectos
	Eficacia en la eliminación de defectos
	Cantidad de veces que un módulo fue probado
	Tamaño del módulo
	Tiempo promedio de corrección de defectos
Soporte a los clientes	Tamaño del back log de defectos
	Tiempo de respuesta en atender los defectos
	Tiempo de resolución de defectos
Herramientas	Soporte de herramientas para procesos propuestos

Tabla 2.3. Métricas de proceso existentes en Ingeniería de Software [Basso,2014].

Para usar las métricas adecuadamente, no es suficiente con medir los atributos cuantitativamente, sino que es necesario tener en cuenta consideraciones como; unidades que se aplican, el tipo de software al que es aplicable y las condiciones en que se deben recoger los datos [López et al., 2016].

2.3. El Factor Humano

Según [Juristo & Moreno, 2013] la ingeniería de software puede considerarse como un proceso social en el que los artefactos (métodos / herramientas / paradigmas) que se utilizarán se ven afectados por la experiencia, el conocimiento y la capacidad de quien los emplea. A diferencia de otras disciplinas que tratan con las leyes de las ciencias naturales, por ejemplo, las leyes de la física no difieren si son utilizadas por un principiante o por un experto, y lo mismo ocurre con una reacción química. Los procesos naturales difieren de los procesos sociales, en que estos últimos son el producto de la intención o conciencia humana. Por lo tanto, una diferencia importante entre la ingeniería de software y otras disciplinas de ingeniería es la importancia del elemento humano. Abdel-Hamid [1989] identifica dos factores que afectan tanto a la calidad como a la productividad del software; (I) las características de la tarea (es decir, la naturaleza compleja de una tarea) y (II) los recursos del equipo (es decir, las habilidades del programador). El impacto del capital humano es tenido en cuenta en métodos de estimación como ser COCOMO [Boehm, 1981] que desde su primera versión identifica factores relacionados con las capacidades del programador en el desarrollo de software, estos factores de ajuste se mantienen en versión actual de método COCOMO II [Boehm et al., 2000]. En muchos estudios se hacen referencia a las métricas que se pueden establecer sobre el código producido con un determinado lenguajes de programación [McCabe, 1976; Halstead, 1977; Lorenz y Kidd, 1994; Rilling y Klemola, 2003; Hernández-López et al., 2013], en el caso de utilizarlas para determinar si un lenguaje es superior a otro, no se debe perder de vista que dichas métricas solo se enfocan en el código resultante pero no tienen en consideración las características del programador que elaboró el código que se está evaluando y por lo tanto se corre el riesgo de sobre calificar o sub calificar un lenguaje por un falencia o virtud del programador.

2.4. Experimentación en Ingeniería de Software

La noción de ingeniería de software, según la definición de IEEE Standard 610.12 es aplicar conocimiento científico al desarrollo, operación y mantenimiento de sistemas de software. La

experimentación en general es una parte importante de dicho conocimiento científico [Juristo & Moreno, 2013]. A pesar de ello, la experimentación en la ingeniería de software no aparenta ser un recurso extensamente utilizado, entre 5453 artículos científicos publicados en 12 revistas y conferencias relevantes entre 1993 y 2002, tan solo 103 artículos (1,9%) presentaban experimentos controlados [Sjoberg et al., 2005]. Según [Genero, M et al., 2014] en la informática en general, y en la ingeniería del software en particular, no se le ha concedido la importancia que se merece a los métodos de investigación, y se han argumentado muchas excusas para no investigar. En su trabajo presentan las excusas más comunes, según Tichy [1998], y sus correspondientes refutaciones (Tabla 2.4.).

Excusas	Refutación
El método científico no es aplicable.	Para entender el proceso de la información, los científicos informáticos deben observar los fenómenos y formular y probar explicaciones.
El nivel de experimentación actual es suficiente.	Comparando con otras ciencias, los científicos informáticos validan un porcentaje mínimo de sus afirmaciones.
Los experimentos tienen un coste muy alto.	Se pueden llevar a cabo experimentos significativos con presupuestos pequeños.
Las demostraciones son suficientes.	Las demostraciones sólo ilustran un potencial pero no demuestran nada.
La experimentación ralentiza el progreso.	Aumentar el porcentaje de artículos con validación significativa es una buena forma de acelerar el progreso.
Las tecnologías cambian demasiado rápido.	Si una cuestión se vuelve irrelevante de forma rápida es que no estaba bien planteada o definida.

Tabla 2.4. Excusas para no investigar en ingeniería del software y sus refutaciones [Tichy, 1998]

La experimentación proporciona una manera sistemática, disciplinada, cuantificable y controlada de evaluar actividades desarrolladas por humanos [Wohlin, Runeson, Höst, & Ohlsson, 2012]. Esta refiere a emparejar con hechos las suposiciones, especulaciones y creencias que abundan en la construcción del software [Juristo & Moreno, 2013]. Enfrentado a una confusa gama de opciones para producir software, los ingenieros de software necesitan pruebas de que un enfoque o técnica particular es realmente mejor que otro.

3. PLANTEAMIENTO DEL PROBLEMA

En este capítulo se describen los problemas relacionados con la evaluación de las características de los lenguajes de programación (sección 3.1), luego se proponen las preguntas de investigación (sección 3.2).

3.1 Descripción del Problema

Como se ha expresado en el (Capitulo 2) en la actualidad existe una gran diversidad de lenguajes de programación; a menudo esto dificulta la tarea de seleccionar el lenguaje que mejor se adapta a las necesidades de un proyecto en particular, ya que determinar cuál es el lenguaje de programación adecuado para desarrollar una solución implica múltiples factores de análisis. Al momento de establecer métricas para determinar si un lenguaje es una mejor alternativa frente a otro, sin perder de vista que dichas métricas no solo se tienen que enfocar en el código resultante sino que también deben de considerar las características del programador que construyó dicho código. Desatender esta situación puede ocasionar la sub calificación o sobre calificación de un lenguaje de programación producto del desempeño de los programadores que los utilizan. Esto último, plantea un problema a la hora de diseñar un experimento cuyo propósito sea determinar el lenguaje de programación a utilizar, ya que el grupo de programadores que utilizan el lenguaje A puede poseer un nivel de conocimientos superior que el grupo de programadores que emplean el lenguaje B o viceversa y esto impactaría en el resultado del experimento. Una posible solución a esta problemática sería contar con un conjunto de programadores que sean capaces de resolver la misma tarea en todos los lenguajes que se requieren evaluar. Más allá de la dificultad que sería reunir a este conjunto de sujetos con los conocimientos necesarios en cada uno de los lenguajes que se pretenden evaluar, esta solución presenta algunas dificultades adicionales que surgen a la hora de diseñar experimentos. Tanto en el trabajo de Juristo y Moreno [2013] como en el de [Wohlin, C. et al., 2012] se considera que los experimentos realizados en el campo de la ingeniería de software son fuertemente influenciados por las características de los sujetos al igual que ocurre en otras ciencias, generalmente conocidas como Ciencias Sociales. Juristo y Moreno [2013] enumeran algunos puntos relacionados con los factores sociales y las características específicas de desarrollo de software que deben tenerse en cuenta al diseñar experimentos.

- Efecto aprendizaje: Si un sujeto tiene que resolver el mismo problema aplicando diferentes lenguajes de programación es muy probable que este aprenda cada vez más sobre el problema y que el último resultado sea mejor que el primero, simplemente porque el sujeto sabe más sobre el problema y no porque el lenguaje de programación sea mejor.
- Efecto de aburrimiento: Los sujetos se aburren o se cansan del experimento y ponen menos esfuerzo e interés a medida que pasa el tiempo.
- Efecto de entusiasmo: Puede suceder que los sujetos que utilizan un lenguaje de programación antiguo no están motivados para hacer un buen trabajo, mientras que los que utilizan un nuevo lenguaje de programación si lo están.
- Efecto de la experiencia: A la hora de realizar un experimento que involucra programadores es de esperar que existan distintos niveles tanto de conocimiento como de habilidad sobre el lenguaje de programación empleado.
- La formalización inconsciente: Surge cuando un mismo sujeto utiliza dos o más lenguajes de programación con diferentes grados de definición o formalidad.
- Efecto de ajuste: El estado emocional de los sujetos participantes se encuentra íntimamente relacionado con el rendimiento que estos tendrán.

A continuación se describen un conjunto de acciones a considerar para controlar los efectos antes enumerados:

- Efecto Aprendizaje: No utilizar al mismo conjunto de sujetos para desarrollar en más de un lenguaje de programación.
- Efecto de aburrimiento: Motivar a los sujetos que ejecutan el experimento de igual manera independientemente del grupo que integren.
- Efecto de entusiasmo: No poner al tanto a los sujetos sobre las hipótesis formuladas o los objetivos del experimento.
- Efecto de la experiencia: Para controlar este efecto se formarán pares experimentales de programadores que sean indistinguibles en lo que refiere a sus conocimientos y habilidades con el lenguaje de programación.
- La formalización inconsciente: Deberán ser considerados aspectos referidos al nivel de conocimiento que cada sujeto tiene en lo que refiere al lenguaje de programación empleado en el momento de formar el par experimental.

- Efecto de ajuste: Se debe de tener en cuenta que todas las instancias del experimento se realicen bajo las mismas condiciones.

Dentro del área de ingeniería de software, es habitual enfrentarse a la necesidad de realizar experimentos con un conjunto reducido de personas las cuales aplican diferentes tratamientos a los objetos de estudio. Realizar comparaciones dentro de pares homogéneos de unidades experimentales no solo aumenta la precisión del análisis sino que también permite controlar gran parte de los efectos no deseados a la hora de realizar un experimento [Juristo & Moreno, 2013].

3.2. Preguntas de Investigación

Este trabajo se desarrolla bajo la hipótesis de que es posible formar pares experimentales homogéneos de programadores y que por lo tanto dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. De esta hipótesis se derivan las siguientes preguntas de investigación: ¿es posible formar pares experimentales indistinguibles en lo que refiere a sus aptitudes como programadores? De ser posible formarlos, ¿Un par experimental integrado por dos programadores que emplean un mismo lenguaje requieren un tiempo similar para resolver la misma tarea?.

4. ASPECTOS ÉTICOS DE LA INVESTIGACIÓN

Aunque la investigación en ingeniería de software no presenta muchas cuestiones éticas, como puede ser en el caso de la medicina o la genética, es necesario tenerlas en cuenta [Genero et al., 2014]. Según Vinson y Singer [2008] las cuestiones éticas planteadas por los métodos empíricos (encuestas, los experimentos, las métricas, los estudios de casos y los estudios de campo) utilizados para investigar tanto los procesos de ingeniería de software como los productos han recibido poca atención en la literatura de la ingeniería de software a pesar de que la popularidad de dichos métodos está en aumento. En (Sección 4.1.) se establece el estado de la cuestión en lo referente a los aspectos éticos en relación a las investigaciones en el campo de la ingeniería de software. Basándose en una extensa investigación documental Vinson y Singer [2002] enumeran los aspectos éticos que deben considerarse en las investigaciones que involucran seres humanos y proponen una serie de principios que serán detallados en la (Sección 4.2.). A pesar de no disponer localmente de indicaciones específicas para las investigaciones en el campo de la ingeniería de software, si se dispone de los criterios establecidos por el CONICET, de carácter general y de áreas específicas, en particular los códigos humanísticos consideran los puntos propuestos por Vinson y Singer [2009] y lo hacen en forma más profunda y detallada que los códigos científico-tecnológicos. Por consiguiente (Sección 4.3.) se enumeran los principios éticos establecidos por el CONICET [2006] y se los aplicará para elaborar las secciones del documento de consentimiento libre e informado (Sección 4.4.).

4.1. Estado de Situación

A partir de revisar los códigos de ética relacionados con la investigación en el campo de la ingeniería de software; Oliveros y Martinez [2012] sostienen que la preocupación por la ética se ha manifestado en distintos trabajos experimentales, pero en pocos trabajos con el objetivo de producir conceptos que permitan guiar la conducta de los investigadores. En su trabajo [Genero, M et al., 2014] hacen referencia a que lamentablemente, ni las agencias de financiación ni las universidades suelen contemplar los aspectos éticos en la investigación en ingeniería del software y que por otro lado, los códigos deontológicos de asociaciones como, por ejemplo, IEEE o de algunas sociedades informáticas específicas, tampoco suelen contemplar los aspectos éticos de la investigación en este campo.

Por su parte Sieber [2001] sostiene que la investigación empírica en el campo de la ingeniería de software con sujetos humanos no es arriesgada, pero que tampoco está exenta de riesgos y considera que si la ética no se tiene en cuenta en la planificación y realización de la investigación, se puede causar daños a algunos de los implicados en la investigación:

- Personas, empresas e investigadores si no se respeta su propiedad intelectual.
- Ingenieros de software que participan o eligen no participar en la investigación.
- Estudiantes a los que se les pide que sirvan como sujetos.

Sieber [2001] señala varios tipos de riesgos involucrados en una investigación. Además, las situaciones particulares son riesgosas en relación con el contexto y las vulnerabilidades particulares de los sujetos involucrados:

- Inconveniencia, como aburrimiento, frustración o pérdida de tiempo por parte de los sujetos. En el caso de los ingenieros de software que sirven como sujetos de investigación, tenga en cuenta que perder su tiempo constituye un gasto para su empleador. En el caso de los estudiantes estos pueden considerar que están mal gastando su tiempo.
- Riesgo psicológico, debido a la preocupación sobre posibles críticas a la forma de trabajar, pérdida de reputación, falta de confidencialidad, la forma en que los resultados de la investigación pueden influir en su carrera. En los estudios, en el lugar de trabajo los empleados pueden creer que el estudio tienen una agenda oculta y que están siendo evaluados. En el caso de los estudiantes estos pueden preocuparse por si su participación (o no participación) en el experimento afectará su calificación y cómo lo hará.
- Riesgo social, como la desaprobación por parte de los pares (colegas) o la estigmatización al brindar cierta información.
- Riesgo económico, además de la pérdida de propiedad intelectual ya mencionada, pérdida de empleo, de oportunidad o de ingresos.
- Riesgo legal, debido a demandas por daños causados a la reputación de la empresa o de individuos

Por todo ello, una de las cuestiones más importantes es la de informar a los sujetos de la naturaleza de la investigación y de sus posibles riesgos, sobre todo por los procedimientos que se seguirán en lo concerniente a la difusión de los resultados, especialmente la privacidad y confidencialidad.

4.2. Criterios establecidos por Vinson y Singer

Vinson y Singer [2008] a partir de analizar los códigos de ética relacionados con las investigaciones que involucran seres humanos descubrieron cuatro principios comunes: consentimiento informado, valor científico, confidencialidad y beneficio.

4.2.1. Consentimiento informado

Estipula que los sujetos potenciales deben ser informados de todos los hechos relevantes sobre un estudio antes de tomar una decisión explícita, libre y bien considerada acerca de considerar su participación. La divulgación es un componente requerido del proceso de consentimiento, ya que permite a los sujetos potenciales evaluar la conveniencia de la participación.

4.2.2. Valor científico

El estudio también debe tener cierto valor científico para llamar a los sujetos humanos a exponerse a riesgos incluso mínimos. Se deben utilizar metodologías apropiadas y pertinentes. Si el estudio no es metodológicamente válido, sus resultados no reflejarán fielmente la realidad y en consecuencia, el estudio no proporcionará ningún beneficio.

4.2.3. Confidencialidad

Los investigadores también deben realizar todos los esfuerzos para mantener la confidencialidad de los datos y la información sensible, para lo cual deben establecer procedimientos para salvaguardar la confidencialidad. Se debe enfatizar la importancia de la confidencialidad a todos los involucrados en el estudio, ya sean investigadores, asistentes de investigación, asignaturas, gerentes o profesores. El análisis de los datos no debe permitir revelar la identidad de los sujetos.

4.2.4. Beneficio

La investigación puede ser beneficiosa para los individuos participantes o para un grupo específico de personas o puede mejorar el conocimiento sobre un tema. En la práctica, esto significa que todos los proyectos de investigación deben someterse a una cuidadosa ponderación del balance riesgo/beneficio. La ponderación de los riesgos, daños y beneficios debe ser positiva para poder proceder con la investigación.

4.3. Criterios Establecidos por el CONICET

A continuación se enumeran los principios éticos que son aplicables a los trabajos de investigación experimental en el campo de la ingeniería de software, dichos principios rigen la relación con las personas que son sujeto de investigación para las ciencias sociales, CONICET [2006]:

4.3.1. Derechos del participante

Se debe respetar la dignidad, la libertad y la autodeterminación del individuo. Las personas que son sujeto de investigación no pueden ser sometidas a perjuicio, riesgo o a cualquier tipo de presión.

4.3.2. Consentimiento libre e informado

Los proyectos de investigación no deben realizarse sin haber obtenido el consentimiento libre e informado de los participantes. Los sujetos de investigación pueden en todo momento interrumpir su participación sin ninguna consecuencia para ellos.

4.3.3. Brindar información

A los sujetos de investigación se les debe proveer toda la información necesaria de tal manera que puedan comprender las consecuencias de participar en el proyecto, el tipo y el propósito de la investigación y las fuentes de financiamiento.

4.3.4. Comunicar el alcance

Los investigadores tienen la responsabilidad de no generar falsas expectativas, comunicando a los sujetos el alcance de la investigación.

4.3.5. Informar resultados

En caso de ser solicitado, los investigadores tienen la obligación de informar a los sujetos de investigación los resultados disponibles en forma apropiada y comprensible.

4.3.6. Confidencialidad

Los investigadores han de respetar la privacidad y están obligados a la confidencialidad de toda información. En particular deben ser cuidadosos con los archivos o listados que identifiquen a los individuos participantes.

4.3.7. Uso de la información

La información no puede ser utilizada sin autorización para otros propósitos, en especial para uso comercial o administrativo.

4.3.8. Respeto

Los investigadores deben tratar con respeto los valores y concepciones de los participantes.

4.4 Criterios para la Elaboración del Consentimiento Libre e Informado

Para elaborar el documento de consentimiento (Anexo A), el cual cumple ampliamente tanto con lo planteado por el CONICET (2006) como por Vinson y Singer (2008), se tomaron como modelo las secciones recomendadas por el Comité de Ética de la Investigación perteneciente a la Organización Mundial de la Salud [OMS, 2011]. A continuación se describen las secciones del documento.

4.4.1. Introducción

En esta sección se describe brevemente quién es el responsable de la investigación y se le explica que se los está invitando a participar y que puedan tomarse su tiempo para reflexionar sobre si quieren participar o no. En este punto es importante aclararle al participante que si no entiende algunas de las palabras o conceptos, pueden hacer preguntas ahora o más tarde.

4.4.2. Objetivo y financiamiento

En esta sección se explica en términos simples el por qué se está haciendo la investigación, cuáles son los objetivos que se plantea, dentro de qué grupo de investigación se enmarca y el modo de financiamiento.

4.4.3. Selección de participantes

En esta sección se indica cuál fue el criterio de selección de los participantes, se busca aclarar por qué este participante ha sido elegido para esta investigación.

4.4.4. Participación voluntaria

En esta sección se indica de manera clara que los participantes pueden optar por participar o no de la investigación y que esto incluye el derecho a retirarse en cualquier momento.

4.4.5. Procedimientos

En esta sección se describen los procedimientos que se seguirán paso a paso, se busca dejar en claro a los participantes qué esperar del estudio y qué se espera de ellos.

4.4.6. Duración

En esta sección se realiza una declaración sobre los compromisos de tiempo de la investigación para con el participante, incluyendo tanto la duración de la investigación y el seguimiento, si es pertinente.

4.4.7. Confidencialidad

En esta sección se explica cómo la investigación mantendrá la confidencialidad de los datos, especialmente con respecto a la información que permite individualizar al participante.

4.3.8. Resultados

En esta sección se le informa al participante de qué manera se compartirán los resultados, es importante dejar en claro si los mismos se compartirán de manera amplia a través de publicaciones y conferencias.

4.3.9. Certificado de consentimiento

Esta sección se redacta en primera persona ya que el sentido de la misma es que el participante exprese su adhesión, mediante una firma, a formar parte de la investigación y deje en claro que dicha decisión la ha tomado en libertad y en pleno conocimiento de la información que el documento contiene.

5. SOLUCIÓN PROPUESTA

La realización de comparaciones dentro de pares homogéneos de unidades experimentales a menudo puede aumentar la precisión del análisis [Juristo y Moreno, 2013]. Existen un gran número de investigaciones en el campo de la ingeniería de software donde se requiere formar pares experimentales de programadores, por ejemplo en el caso de buscar comparar los tiempos de desarrollo de una misma aplicación programada en diferentes lenguajes de programación por distintas personas (consideradas homogéneas), a efectos de medir si el lenguaje empleado influye en la productividad. Pero al igual que en la mayoría de las actividades humanas, el rendimiento individual en el desarrollo de software varía considerablemente de una persona a otra y se deben articular mecanismos para que dichas variaciones no afecten los resultados del estudio.

A la hora de definir como formar pares homogéneos de programadores se debe de tener en cuenta lo expresado por [Campbell et al., 1993], en su trabajo sostiene que muchos son los factores que afectan el rendimiento de un individuo de manera indirecta, pero sólo hay tres determinantes directos del rendimiento: conocimiento, habilidad y motivación. Este protocolo tiene por propósito la formación de pares experimentales homogéneos, contemplando tanto los factores de conocimiento como los de habilidades y dando por sentado que todos los participantes a caracterizar no presentan diferencias significativas en lo que respecta a la motivación. Etimológicamente, el término motivación procede del latín *motus*, que se relaciona con aquello que moviliza a la persona para ejecutar una actividad. Vernon (1973) define la motivación como una fuerza interna que emerge, regula y sostiene las acciones de los seres humanos. Santrock (2002) indica que existen tres perspectivas fundamentales respecto de la motivación: la conductista la cual destaca el papel de las recompensas en la motivación, la humanista la cual subraya las capacidades del ser humano para desarrollarse y la cognitiva la cual enfatiza en el poder del pensamiento. Desde la perspectiva cognitiva, Ajello (2003) señala que la motivación debe ser entendida como la trama que sostiene el desarrollo de aquellas actividades que son significativas para la persona. Según Vroom (1964), la motivación es el resultado de tres variables: valencia, expectativas e instrumentalidad.

- La valencia se refiere al valor que la persona aporta a cierta actividad, el deseo o interés que tiene en realizarla.
- Las expectativas se definen como las creencias sobre la probabilidad de que un acto irá seguido de un determinado resultado.

- La instrumentalidad se refiere a la consideración que la persona hace respecto de que si logra un determinado resultado, este servirá de algo.

Entendiendo que la motivación es importante, el protocolo propuesto establece que para ser aplicado los participantes no deben presentar diferencias significativas en lo que respecta a la motivación. Para lograr que la motivación no presente diferencias significativas entre los participantes de los casos de estudio en los cuales se desee aplicar el protocolo de formación de pares experimentales de programadores y teniendo en cuenta que si una persona no se siente capaz o no tiene interés o piensa que el esfuerzo realizado no va a tener repercusión, entonces no tendrá motivación, se deberá procurar que los participantes:

- Sientan interés en participar
- Comprendan el propósito de realizar las tareas propuestas
- Se sientan capaces de realizar las tareas propuestas

Para lograr que la motivación no presente diferencias significativas entre los participantes de los casos de estudio que forman parte de este trabajo se procuró que estos:

- Estén notificados del propósito del estudio
- Estén notificados del porque ellos forman parte del mismo
- Participen de forma completamente voluntaria
- Manifiesten conocer alguno de los lenguajes de programación
- Sean libres de contestar o no cualquiera de las preguntas
- Sean libres de realizar o no cualquiera de los ejercicios
- Sean libres de retirarse en cualquier momento
- No inviertan más que el tiempo acordado para realizar el estudio

En la (Sección 5.1.) se identifican las formas utilizadas para clasificar a los programadores en otros estudios de investigación experimental, en la (Sección 5.2.) se construyen las versiones preliminares de las herramientas de caracterización que se utilizará en esta investigación, en la (Sección 5.3.) se establece un mecanismo que permite formar pares experimentales homogéneos de programadores en base a la caracterización realizada y en la (Sección 5.4.) se lleva adelante la validación del protocolo mediante la realización de una prueba piloto, la cual permite obtener versión mejorada.

5.1 Formas de determinar la experiencia en programación

[Feigenspan et al., 2012] realizan un trabajo documental en el cual se analizan 161 publicaciones y se identifican nueve formas utilizadas por los investigadores para determinar la experiencia que un programador posee, los autores definen la experiencia como la cantidad de conocimientos adquiridos respecto al desarrollo de programas. En el (Gráfico 5.1) se pueden ver los resultados representados y a continuación se enumeran las nueve formas para determinar la experiencia en programación:

1. Años: En muchos documentos (47), los años que un participante ha estado programando en general o en una empresa o en cierto lenguaje se usó para determinar la experiencia de programación.
2. Educación: La educación de los participantes se utilizó para indicar su experiencia en diecinueve (19) de los artículos revisados. Incluyendo información sobre el nivel de educación alcanzado (pregrado, grado, posgrado, etc.) o las calificaciones de los cursos.
3. Auto Estimación: En doce (12) trabajos, se pidió a los participantes que estimaran su propia experiencia.
4. Cuestionario específico: En nueve (9) trabajos los autores aplicaron un cuestionario para evaluar la experiencia de programación.
5. Tamaño: El tamaño de los programas que los participantes habían escrito fue utilizado como indicador en seis (6) artículos.
6. Examen: En tres (3) trabajos se realizó un examen de programación para evaluar la experiencia de los participantes.
7. Supervisor: En dos (2) trabajos, en los que los programadores profesionales fueron utilizados como participantes, un supervisor fue el encargado de estimar su experiencia.
8. No especificado: A menudo, los autores afirman que midieron la experiencia de programación, pero no especificaron como se realizó la medición. Este fue el caso en treinta y nueve (39) trabajos.
9. No controlada: La experiencia de programación no se mencionó en absoluto en cuarenta y cinco (45) trabajos, lo que amenaza la validez de los experimentos correspondientes.

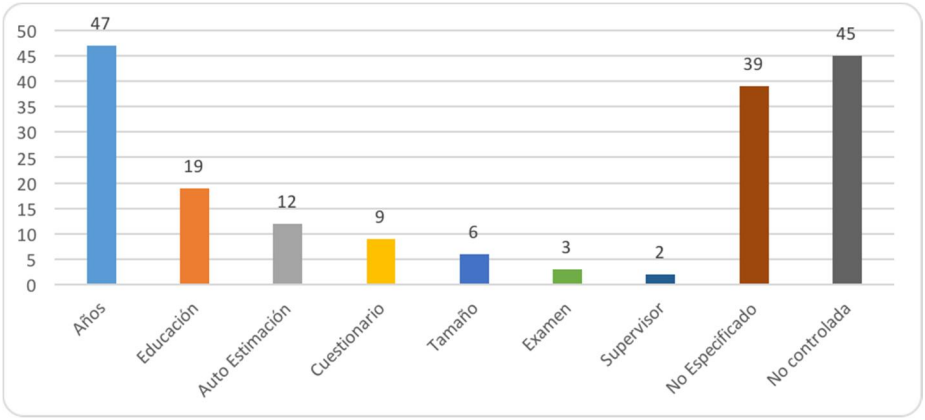

Gráfico 5.1 Formas de determinar la experiencia en programación [Feigenspan et al., 2012]

5.2. Caracterización de los Programadores

En su investigación [Kruger y Dunning, 1999] determinaron que las personas menos competentes tienden a sobrestimar su habilidad al no tener conocimiento suficiente para reconocer sus propias limitaciones y que también es común que personas más preparadas tiendan a subestimar sus logros y sus competencias. Se requiere desarrollar un método de caracterización que no se base en la percepción que tiene cada individuo acerca de sus habilidades como programador, considerando la experiencia con proyectos de desarrollo de software, los conocimientos de lenguajes de programación, los conocimientos de algoritmos, los conocimientos del ambiente de desarrollo y otras habilidades relacionadas con la persona que pueden tener un efecto en el tiempo que un participante necesita para desarrollar una solución informática [Wilking, D. et al., 2008].

Esta caracterización persigue como único objetivo calificar un conjunto de aptitudes del programador a solo efecto de poder encontrar parejas de programadores que se puedan considerar homogéneas. El propósito que tiene la caracterización es asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. Se optó por analizar un amplio conjunto de aptitudes del programador por entender que realizar una caracterización basada tan solo en unos pocos criterios puede incurrir en graves errores.

Para la construcción de las herramientas de caracterización se ha tenido en cuenta el procedimiento general de construcción de un instrumento de medición propuesto por [Sampieri, R. et al., 2010)] y el método para la definición de métricas válidas propuesto por [Genero, M. et al., 2014], adaptando dichos procedimientos a la necesidad de este trabajo. Los pasos a seguir para la construcción de dichos instrumentos se plantean en la (Figura 5.1.). En la (Sección 5.2.1.) se desarrollan los pasos tres primeros pasos de la construcción de la herramienta CILP (Caracterización Independiente del

Lenguaje de Programación) y en la (Sección 5.2.2.) se desarrollan los tres primeros pasos de la construcción de la herramienta CDLP (Caracterización Dependiente del Lenguaje de Programación).

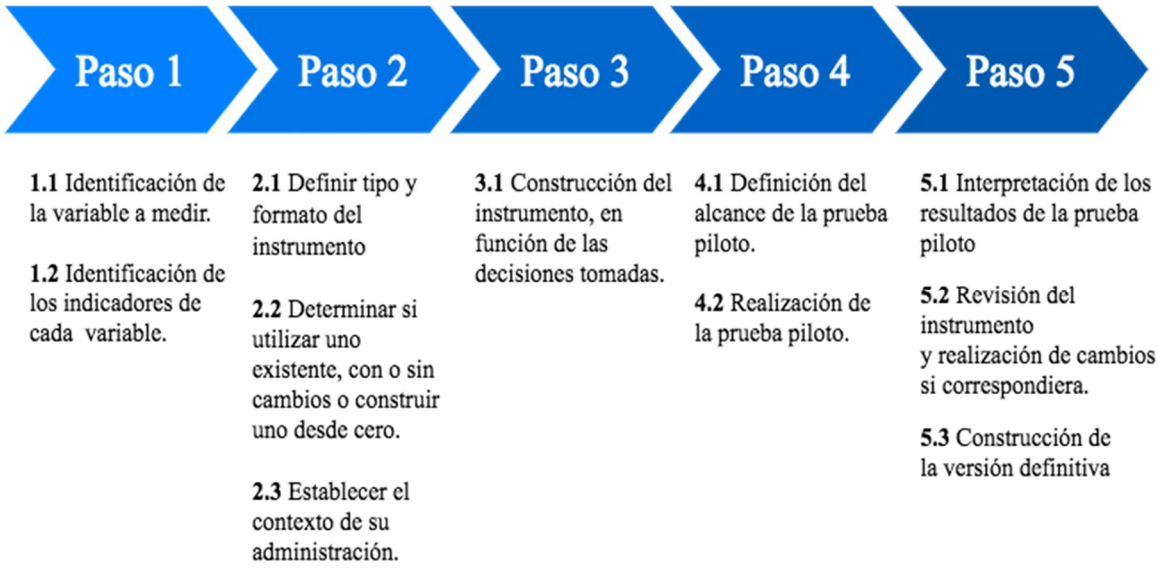

Figura 5.1. Pasos a seguir para la construcción de la herramienta [Sampieri et al., 2010].

5.2.1. Construcción de la Herramienta de Caracterización Independiente del Lenguaje de Programación

El objetivo de la herramienta es calificar el conjunto de aptitudes del programador que son independientes del lenguaje de programación.

5.2.1.1. Identificación de la variable a medir e indicadores [Paso 1]

Sampieri et al. [2010] consideran que al comenzar a desarrollar el o los instrumentos de medición, vale la pena una última reflexión que permita cerciorarse de cuáles son las variables y algunas otras precisiones. Para lo cual sugiere contestar las una serie de preguntas a las que se les ha dado respuesta en la (Tabla 5.1.).

Preguntas	Respuestas
¿Qué se va a medir?	Características de los programadores que son independientes a un lenguaje de programación en particular.
¿Qué o quiénes va a ser medidos?	Programadores de sistemas informáticos.
¿Cuándo?	La herramienta se utilizará antes de formar los pares experimentales ya que dicha formación dependerá, en parte, de las características relevadas por esta herramienta.

¿Dónde?	En cualquier investigación que se encuentre ante la necesidad de formar pares experimentales de programadores.
¿Qué tipo de datos queremos obtener?	Las características de un programador que son independientes a un lenguaje de programación en particular.
¿Nuestro propósito al recolectar los datos es?	Contar con indicadores que nos permitan formar pares experimentales de programadores.

Tabla 5.1. Preguntas para la identificación de las variables y otras precisiones

A continuación se presentan los dominios del contenido de la variable (dimensiones), los indicadores de cada dimensión y la nomenclatura propuesta de las dimensiones. En la (Tabla 5.1) se presenta la variable, sus dimensiones, la nomenclatura propuesta para cada dimensión y los indicadores de la CILP (Caracterización Independiente del Lenguaje de Programación).

Variable a medir	Dimensión	Nomenclatura de la Dimensión	Indicador
Características independientes del lenguaje de programación	*Nivel de formación.*	*CILP1*	Refiere al nivel alcanzado por el participante en su educación ya sea formal o informal.
	Experiencia	*CILP2*	Refiere a los años de experiencia como programador y a la cantidad de lenguajes que manifiesta conocer.
	Comprensión de una Especificación	*CILP3*	Capacidad que tiene el programador para comprender una especificación simple y el tiempo insumido.
	Comprensión de Pseudocódigo	*CILP4*	Capacidad que tiene el programador de interpretar el funcionamiento de bloques de pseudocódigo y el tiempo insumido. Estará compuesto por un enunciado, en el cual el participante deberá poder identificar cuatro aspectos (Datos de entrada, datos de salida, estructuras condicionales y estructuras repetitivas).
	Capacidad Algorítmica	*CILP5*	Capacidad que tiene el programador en desarrollar una solución con pseudocódigo y el tiempo que emplea en realizar dicha solución.

Tabla 5.2. Variable, sus dimensiones, la nomenclatura propuesta para cada dimensión y sus indicadores

Para la construcción de los ítems y de las opciones de respuesta de las dimensiones, se decidió plantear una tabla para cada una de las dimensiones. Resultando las tablas que se presentan a continuación:

Dimensión	Ítems	Opciones de respuesta
Nivel de formación. (CILP1)	De las siguientes alternativas seleccione la que corresponde al nivel de estudios del que posee título oficial. (CILP1.1)	Primaria Secundaria no técnica Secundaria técnica Terciaria Universitaria de pregrado Universitaria de grado Universitaria de posgrado
	¿Cuántas materias o cursos de programación ha realizado hasta el momento? (CILP1.2)	Ninguno Uno Dos Tres Cuatro Entre cinco y diez Más de diez

Tabla 5.3. Dimensión Nivel de formación, los ítems y las opciones de respuesta del CILP

Dimensión	Ítems	Opciones de respuesta
Experiencia. (CILP2)	¿Cuántos años de experiencia tiene como programador? (CILP2.1)	Menos de uno Uno Dos Tres Cuatro Entre cinco y diez Más de diez
	¿En cuántos lenguajes de programación considera tener un nivel regular o superior? (CILP2.2)	Ninguno Uno Dos Tres Cuatro Entre cinco y diez Más de diez

Tabla 5.4. Dimensión Experiencia, los ítems y las opciones de respuesta del CILP

Dimensión	Ítems	Opciones de respuesta
Comprensión de una Especificación. (CILP3)	Indique el resultado que se espera obtener al ejecutar el algoritmo.	La cantidad de ocurrencias del número más grande. El valor del número más grande. La posición de cada uno de los números con valor igual al número más grande. La posición de la primera ocurrencia del número con valor igual al número más grande. Ninguna de las anteriores
	Indique si existe la necesidad de solicitar algún dato al usuario.	NO. SI, se solicita la cantidad de líneas del archivo. SI, se solicita el nombre del archivo. SI, se solicita la ruta al archivo.
	Indique si existe la necesidad de utilizar una estructura condicional.	NO SI
	Indique si existe la necesidad de utilizar una estructura repetitiva.	NO SI

Tabla 5.5. Dimensión Comprensión de una Especificación, los ítems y las opciones de respuesta del CILP

Dimensión	Ítems	Opciones de respuesta

Comprensión de Pseudocódigo. (CILP4)	Suponiendo el siguiente bloque de pseudocódigo :
	
	FOR X = 1 to 100
	  IF X%10 == 0 THEN
	  PRINT X;
	  X=X*2;
	  END IF
	END FOR
	
	¿Cuál es la salida por pantalla?
		Números que son múltiplo de diez entre uno y cien
		
		Números que no son múltiplo de diez entre uno y cien
		
		Números que son múltiplo de diez entre uno y cien al cuadrado
		
		Números que son múltiplo de diez entre uno y cien al cuadrado
		
		Ninguna de las anteriores.

	Suponiendo el siguiente bloque de pseudocódigo :
	
	NUM X=49;
	NUM Z=0;
	WHILE X < 50
	  PRINT X;
	  IF Z== 5 THEN
	  X=X+1;
	  END IF
	  Z=Z+1;
	END WHILE
	
	¿Cuál es la salida por pantalla?
		Números entre cero y cuarenta y nueve.
		
		Solo el número cuarenta y nueve.
		
		El número cuarenta y nueve en cinco ocasiones
		
		El número cuarenta y nueve en seis ocasiones
		
		Ninguna de las anteriores.

Tabla 5.6. Dimensión Comprensión de Pseudocódigo, los ítems y las opciones de respuesta del CILP

Dimensión	Ítems	Opciones de respuesta
Capacidad Algorítmica. (CILP5)	Luego de ser completados los lugares en blanco el programa deberá permitir validar los datos de acceso de un usuario al sistema. El usuario tendrá como nombre de acceso 'admin' y como contraseña '2357'. El programa deberá dar como máximo tres oportunidades. Si el usuario ingresa un nombre de cuenta inválido el mensaje a mostrar será 'USUARIO NO REGISTRADO'. Si ingresa bien el nombre de usuario y la contraseña es incorrecta el mensaje será 'CONTRASEÑA INVALIDA'. Si tanto el nombre como la contraseña son correctos el mensaje será 'BIENVENIDO'. En el caso de exceder la cantidad de intentos el programa deberá finalizar.	El participante dispondrá de un conjunto de opciones para completar cada uno de los lugares en blanco.
	Escriba en pseudocódigo un programa que calcule el sueldo neto de un trabajador quien cobra según las horas trabajadas. El cálculo se realiza de la siguiente forma: - El valor de la hora es $100 - Las primeras 160 horas a una tarifa fija. - Las horas extras se pagan a un 50% más de la tarifa fija. - Los impuestos a deducir de los trabajadores varían según el sueldo mensual. Si el sueldo supera los $15000 el trabajador pagara un 10% de impuesto. El programa deberá solicitar al usuario la cantidad de horas trabajadas e indicar por pantalla el sueldo neto que el trabajador percibirá por sus servicios.	El participante deberá escribir un pseudocódigo que satisfaga el requerimiento.

Tabla 5.7. Dimensión Capacidad Algorítmica, los ítems y las opciones de respuesta del CILP

5.2.1.2. Decisión del tipo y formato del instrumento y su contexto de administración [Paso 2]

Se utilizará el procedimiento de recolección de datos mixto conformado por el cuestionario CILP y luego por una entrevista. Tanto las preguntas como los ejercicios se entregan a los participantes en forma de cuestionario. Con el propósito de minimizar los errores de caracterización, una vez que el participante haya contestado el cuestionario, se desarrollará una entrevista individual donde el entrevistador realizará preguntas a efectos de que el participante justifique sus respuestas y ante la correcta justificación el entrevistador dará la misma como válida.

Respecto al contexto de administración será una sala provista de una computadora por programador ya que la primera instancia, la que refiere al cuestionario, es auto-administrada permitiendo llevar esta acción de forma individual o grupal, para continuar con la instancia de entrevista de forma individual.

5.2.1.3. Construcción de la primera versión del Cuestionario [Paso 3]

Algunas cuestiones referidas al procedimiento serán explicadas por el investigador responsable, como por ejemplo: el tiempo aproximado de respuesta, el procesamiento de los cuestionarios, la confidencialidad del manejo de la información, etc. Teniendo en cuenta que para llevar adelante el relevamiento de alguna de las variables se requiere utilizar pseudocódigo se comenzará por definir las reglas que el mismo deberá seguir (Sección 5.2.1.3.1.) para luego listar los ítems que formarán parte del cuestionario (Sección 5.2.1.3.2.).

5.2.1.3.1. Pseudocódigo a ser Utilizado

Loyarte & Novara [2009] definen al pseudocódigo como un falso lenguaje, ya que tiene normas de estructura de un lenguaje de programación pero se encuentra desarrollado para que pueda ser leído por una persona y no interpretado por una máquina. A continuación se definirán las palabras reservadas que tendrá el pseudocódigo:

- **NUM**: Define una variable numérica.
- **STR**: Define una variable del tipo cadena de caracteres.
- **READ**: Lee un dato ingresado por el usuario.
- **PRINT**: Imprime por pantalla una variable.
- **WHILE**: Bucle que es ejecutado mientras se cumpla la condición.
- **IF-THEN-ELSE**: Establece una estructura condicional.
- **REPEAT-UNTIL**: Bucle el cual evalúa la condición al final.
- **FOR A=0 TO B**: Bucle que se ejecuta hasta que A iguala a B.

- **BREAK**: Interrumpe la ejecución del bucle.
- **CONTINUE**: Interrumpe la ejecución de una iteración de un bucle, pero, no la ejecución del bucle en sí.
- **OPERADORES MATEMÁTICOS**
  - \+ : Suma
  - \- : Resta
  - * : Multiplicación
  - / : División
  - % : Resto división entera

- **OPERADORES RELACIONALES**
  - > : Mayor
  - >= : Mayor o igual
  - < : Menor
  - <= : Menor o igual
  - == : Igual
  - != : Diferente

A continuación se realiza un programa en el pseudocódigo propuesto el cual tiene por propósito mostrar el uso de las palabras reservadas (Figura 5.2.). En él, se recorren todas las posiciones de una matriz verificando su contenido, en el caso de ser cero se imprime por pantalla la letra 'X' caso contrario un el símbolo '-', una vez finalizado el recorrido de la matriz se le consulta al usuario si desea continuar y en el caso de responder afirmativamente se repite el proceso.

```
NUM X
NUM Y
NUM tablero[10][10]
STR SEGUIR = 'S'
WHILE SEGUIR == 'S'
        FOR X = 1 to 10
                FOR Y = 1 to 10
                        IF tablero[X][Y] == 0 THEN
                                PRINT "-"
                        ELSE
                                PRINT "X"
                        END IF
                END FOR
        END FOR
        PRINT "PARA SEGUIR PULSE S"
        READ SEGUIR
END WHILE
```

Figura 5.2 Ejemplo de Aplicación Pseudocódigo Propuesto

5.2.1.3.2. Cuestionario de Caracterización Independiente del Lenguaje de Programación

A continuación se enumeran los ítems que forman parte del cuestionario que permite relevar las variables de caracterización que son independientes del lenguaje de programación.

a) De las siguientes alternativas seleccione la que corresponde al nivel educativo máximo del que posee título oficial
   - Primaria
   - Secundaria no técnica
   - Secundaria técnica
   - Terciaria
   - Universitaria de pregrado
   - Universitaria de grado
   - Universitaria de posgrado

b) ¿Cuántas materias o cursos de programación ha realizado hasta el momento?
   - Ninguno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

c) ¿Cuántos años de experiencia tiene como programador?
   - Menos de uno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

d) Suponiendo la siguiente escala de nivel que un programador tiene en un lenguaje de programación (Muy Malo, Malo, Regular, Bueno y Muy Bueno) ¿En cuántos lenguajes de programación considera tener un nivel regular o superior?
   - Menos de uno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

e) Suponiendo que se desea escribir en pseudocódigo un algoritmo capaz de procesar un archivo de texto y en cada línea del mismo se encuentra un número comprendido en el rango [0 - 99]. El algoritmo deberá recorrer el archivo a efectos de informar por pantalla en qué posición se encuentra el número más alto, en el caso de repetirse dicho numero en más de una línea se deberá de informar la primer ocurrencia. La ruta a dicho archivo se le solicitará al usuario al inicio del programa.

1. Indique el resultado que se espera obtener al ejecutar el algoritmo
   - La cantidad de ocurrencias del número más grande.
   - El valor del número más grande.
   - La posición de cada uno de los números con valor igual al número más grande.
   - La posición de la primera ocurrencia del número con valor igual al número más grande.
   - Ninguna de las anteriores
2. Indique si existe la necesidad de solicitar algún dato al usuario y en tal caso cual
   - NO.
   - SI, se solicita la cantidad de líneas del archivo.
   - SI, se solicita el nombre del archivo.
   - SI, se solicita la ruta al archivo.
3. Indique si existe la necesidad de utilizar una estructura condicional
   - NO.
   - SI.
4. Indique si existe la necesidad de utilizar una estructura repetitiva
   - NO.
   - SI.

f) Suponiendo el siguiente bloque de pseudocódigo :

```
FOR X = 1 to 100
        IF X%10 == 0 THEN
                PRINT X;
                X=X*2;
        END IF
END FOR
```

¿Cuál es la salida por pantalla?

- Números que son múltiplo de diez entre uno y cien
- Números que no son múltiplo de diez entre uno y cien
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Ninguna de las anteriores.

g) Suponiendo el siguiente bloque de pseudocódigo :

```
NUM X=49;
NUM Z=0;
WHILE X < 50
        PRINT X;
        IF Z== 5 THEN
                X=X+1;
        END IF
        Z=Z+1;
END WHILE
```

¿Cuál es la salida por pantalla?

- Números entre cero y cuarenta y nueve.
- Solo el número cuarenta y nueve.
- El número cuarenta y nueve en cinco ocasiones
- El número cuarenta y nueve en seis ocasiones
- Ninguna de las anteriores.

h) Luego de ser completados los lugares en blanco de la (Figura 5.3) el programa deberá permitir validar los datos de acceso de un usuario al sistema. El usuario tendrá como nombre de acceso ‘admin’ y como contraseña ‘2357’. El programa deberá dar como máximo tres oportunidades. Si el usuario ingresa un nombre de cuenta inválido el mensaje a mostrar será ‘USUARIO NO REGISTRADO’. Si ingresa bien el nombre de usuario y la contraseña es incorrecta el mensaje será ‘CONTRASEÑA INVALIDA’. Si tanto el nombre como la contraseña son correctos el mensaje será ‘BIENVENIDO’. En el caso de exceder la cantidad de intentos el programa deberá finalizar.

```
1  NUM intento = 0;
2  STR usuario;
3   0  password;
4        1
5    2   (intento  3  3)
6  {
7       usuario =    4    ;
8       password = READ();
9
10      IF( usuario 5 "admin")
11      {
12          PRINT "USUARIO NO REGISTRADO";
13          .   6   ++;
14
15      }
16      7
17      {
18          IF( contraseña  8  "2357")
19          {
20              registrado = 1;
21              BREAK;
22          }
23          PRINT "CONTRASEÑA INVALIDA"
24          intento 9 ;
25      }
26 }
27 IF( registrado 10  1)
28 {
29      PRINT "BIENVENIDO"
30 }
```

Figura 5.3. Ejercicio a completar

i) Escriba en pseudocódigo un programa que calcule el sueldo neto de un trabajador quien cobra según las horas trabajadas. El cálculo se realiza de la siguiente forma:

- El valor de la hora es $100
- Las primeras 160 horas a una tarifa fija.
- Las horas extras se pagan a un 50% más de la tarifa fija.
- Los impuestos a deducir de los trabajadores varían según el sueldo mensual. Si el sueldo supera los $15000 el trabajador pagara un 10% de impuesto.

El programa deberá solicitar al usuario la cantidad de horas trabajadas e indicar por pantalla el sueldo neto que el trabajador percibirá por sus servicios.

5.2.2 Construcción de la Herramienta de Caracterización Dependiente del Lenguaje de Programación

El objetivo de la herramienta es calificar el conjunto de aptitudes del programador que son dependientes del lenguaje de programación.

5.2.2.1. Identificación de la variable a medir e indicadores [Paso 1]

Sampieri et al. [2010] considera que al comenzar a desarrollar el o los instrumentos de medición, vale la pena una última reflexión que permita cerciorarse de cuáles son las variables y algunas otras precisiones. Para lo cual sugiere contestar una serie de preguntas a las que se les ha dado respuesta en la (Tabla 5.1).

Preguntas	Respuestas
¿Qué se va a medir?	Características de los programadores que son dependientes del lenguaje de programación que expresa conocer.
¿Qué o quiénes va a ser medidos?	Programadores de sistemas informáticos.
¿Cuándo?	La herramienta se utilizara antes de formar los pares experimentales ya que dicha formación dependerá, en parte, de las características relevadas por esta herramienta.
¿Dónde?	En cualquier investigación que se encuentre ante la necesidad de formar pares experimentales de programadores.
¿Qué tipo de datos queremos obtener?	Las características de un programador que son dependientes a un lenguaje de programación en particular.
¿Nuestro propósito al recolectar los datos es?	Contar con indicadores que nos permitan formar pares experimentales de programadores.

Tabla 5.8. Preguntas para la identificación de las variables y otras precisiones

A continuación se presentan los dominios del contenido de la variable (dimensiones), los indicadores de cada dimensión, y la nomenclatura propuesta de las dimensiones. En la (Tabla 5.1) se presenta la variable, sus dimensiones, la nomenclatura propuesta para cada dimensión y los indicadores de la CDLP (Caracterización Independiente del Lenguaje de Programación).

Variable a medir	**Dimensión**	**Nomenclatura de la Dimensión**	**Indicador**
Características dependientes del lenguaje de programación	*Lenguaje Elegido*	*CDLP1*	Refiere al lenguaje de programación que el participante expresa dominar con mayor fluidez y a los años de experiencia que tiene con este.
	Conocimiento Teórico	*CDLP2*	Refiere al nivel de conocimientos sobre aspectos teóricos del lenguaje de programación que el participante posee.
	Comprensión de Código Fuente	*CDLP3*	Capacidad que tiene el programador de interpretar el funcionamiento de bloques de código y el tiempo insumido en ello.

Tabla 5.9. Variable, sus dimensiones, la nomenclatura propuesta para cada dimensión y sus indicadores

Para la construcción de los ítems y de las opciones de respuesta de las dimensiones, se decidió plantear una tabla para cada una de las dimensiones. Teniendo en cuenta que las dimensiones dos y tres varían según el lenguaje elegido, se decidió por desarrollar dichas dimensiones tanto para el lenguaje C como para el lenguaje C#. Resultando las tablas que se presentan a continuación:

Dimensión	Ítems	Opciones de respuesta
Lenguaje Elegido. (CDLP1)	¿Cuál es el lenguaje de programación que domina con mayor fluidez? (CDLP1.1)	C C#
	¿Cuántos años de experiencia tiene con el lenguaje de programación elegido?(CDLP1.2)	Menos de uno Uno Dos Tres Cuatro Entre cinco y diez Más de diez
	¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional? (CDLP1.3)	Menos de uno Uno Dos Tres Cuatro Entre cinco y diez Más de diez

Tabla 5.10. Dimensión Lenguaje Elegido, los ítems y las opciones de respuesta del CDLP

Dimensión	Ítems	Opciones de respuesta
Conocimiento Teórico del Lenguaje C. *(CDLP2_C)*	Relacione el nombre de la función con la descripción de la misma.	scanf ___ strcmp ___ realloc ___ sizeof ___ 1. Lee exclusivamente strings de la entrada estándar. 2. Compara dos cadenas de caracteres. 3. Redimensiona memoria previamente reservada con malloc. 4. Retorna la longitud de un array. 5. Copia una cadena de caracteres en otra. 6. Lee datos formateados de la entrada estándar. 7. Reserva memoria de manera dinámica. 8. Retorna el tamaño en bytes ocupado por un tipo de dato.
	Relacione la palabra reservada con la descripción de su propósito.	define ___ continue ___ typedef ___ 1. Sustituye la cadena de token por cada aparición del identificador. 2. Causa la inmediata salida de una estructura de control. 3. Da un nombre nuevo a cualquier tipo de datos. 4. Declara una variable indicando que su valor es inalterable. 5. Produce un salto en la ejecución de código, al pie del bloque que lo contiene. 6. Define un nuevo tipo de dato.
	¿Qué es y para que se utiliza una estructura?	Respuesta abierta.
	¿Qué es un puntero?	Respuesta abierta.
	¿En una función a que se denomina pasaje por valor y pasaje por referencia?	Respuesta abierta.

Tabla 5.11. Dimensión Conocimiento Teórico del Lenguaje C, los ítems y las opciones de respuesta del CDLP

Dimensión	Ítems	Opciones de respuesta

Comprensión de Código Fuente del Lenguaje C. *(CDLP3_C)*	¿Si la variable op tiene valor 'H' cuál es la salida por pantalla del siguiente fragmento de código?
	
	switch(op)
	{
	case 'H' : printf("Hola ");
	case 'J' : printf("Juan ");
	case 'C' : printf("Carlos ");
	break;
	}
		Hola
		Hola Juan
		Hola Juan Carlos
		Ninguna de las anteriores

	¿Cuál es la salida por pantalla del siguiente fragmento de código?
	
	int main()
	{
	int data[5], i;
	for(i = 5; i > 0; i--)
	*(data + i) = i;
	for(i = 0; i < 5; i++)
	printf("%d-", *(data + i));
	return 0;
	}
		1-2-3-4-5-
		5-4-3-2-1-
		0-1-2-3-4-
		4-3-2-1-0-
		Ninguna de las anteriores

	¿Cuál es el propósito del siguiente bloque de código?
	
	int num=50, i=1;
	while(i<=num){
	if(!(num % i)){
	printf("%d", i);
	}
	i++;
	}
		Muestra números pares desde el 1 al 50
		Muestra los divisores de 50
		Muestra números impares desde el 1 al 50
		Muestra los números que no son divisores de 50
		Error de compilación
		No muestra nada

	A partir del siguiente código indique qué valor se muestra por la consola.
	
	int *p, *q;
	int x = 4;
	p = &x;
	q = p;
	*q += 3;
	printf("%d", *p);
		Muestra el número 12
		Muestra el número 4
		Muestra la dirección de x
		Error de compilación
		Muestra el número 7
		No es posible determinar el valor mostrado

Tabla 5.12. Dimensión Comprensión de Código Fuente del Lenguaje C, los ítems y las opciones de respuesta del CDLP

Dimensión	Ítems	Opciones de respuesta
Conocimiento Teórico del Lenguaje C#. *(CDLP2_C#)*	Relacione el nombre de la función con la descripción de la misma.	Console ___ Object.Equals ___ Int32.TryParse ___ List.Count ___ 1. Compara si dos objetos hacen referencia a una misma instancia 2. Retorna la cantidad de ítems de una lista 3. Intenta convertir un dato a entero. Si lo logra retorna el número convertido. 4. Compara dos objetos por su valor 5. Representa los flujos de salida para las aplicaciones de consola. 6. Intenta convertir un dato a entero. Si lo logra retorna true caso contrario retorna false. 7. Retorna el tamaño en bytes ocupado por la lista 8. Representa los flujos de entrada, salida y error estándar para las aplicaciones de consola.
	Relacione la palabra reservada u operador con la descripción de su propósito.	using ___ break ___ : (Ej. C1:C2) ___ 1. Importa una clase. 2. Produce un salto en la ejecución de código, al pie del bloque que lo contiene. 3. Define una relación de herencia, donde C1 hereda de C2. 4. Importar espacios de nombres (namespaces). 5. Define una relación de herencia, donde C2 hereda de C1. 6. Causa la inmediata salida de una estructura de control.
	¿Qué diferencia existe entre un método abstract y uno virtual?	Respuesta abierta.
	¿Qué se puede definir dentro de una interfaz?	Respuesta abierta.
	Si se define class A : B, C ¿Es esto correcto? De serlo, ¿De qué tipo deberían ser B y C?	Respuesta abierta.

Tabla 5.13. Dimensión Conocimiento Teórico del Lenguaje C#, los ítems y las opciones de respuesta del CDLP

Dimensión	Ítems	Opciones de respuesta

Comprensión de Código Fuente del Lenguaje C#. *(CDLP3_C#)*	¿Si la variable op tiene valor 'H' cuál es la salida por pantalla del siguiente fragmento de código?
	
	switch(op){
	case "H":
	Console.WriteLine("Hola ");
	case "J":
	Console.WriteLine("Juan ");
	case "C":
	Console.WriteLine("Carlos ");
	break;
	}
		Hola
		Hola Juan
		Hola Juan Carlos
		Ninguna de las anteriores

	¿Cuál es la salida por pantalla del siguiente fragmento de código?
	
	static int main(){
	int max = 5;
	int[] data = new int[max];
	int i;
	
	for (i = 5; i > 0; i--)
	data[max-i] = i;
	for (i = 0; i < 5; i++)
	Console.WriteLine("{0}-", data[i]);
	return 0;
	}
		1-2-3-4-5-
		5-4-3-2-1-
		0-1-2-3-4-
		4-3-2-1-0-
		Ninguna de las anteriores

	¿Cuál es el propósito del siguiente bloque de código?
	
	int num = 50, j = 1;
	while(j <= num){
	if((num % j) == 0){
	Console.WriteLine("{0}-", j);
	}
	j++;
	}
		Muestra números pares desde el 1 al 50
		Muestra los divisores de 50
		Muestra números impares desde el 1 al 50
		Muestra los números que no son divisores de 50
		Error de compilación
		No muestra nada

	A partir del siguiente código indique qué valor se muestra por la consola.
	
	Clase1 c1 = new Clase1 (12);
	Clase1 c2 = c1;
	c1.Dato = 7;
	c2.Dato = 4;
	Console.WriteLine(c1.Dato);
		Muestra el número 12
		Muestra el número 4
		Muestra la dirección de x
		Error de compilación
		Muestra el número 7
		No es posible determinar el valor mostrado

Tabla 5.14. Dimensión Comprensión de Código Fuente del Lenguaje C#, los ítems y las opciones de respuesta del CDLP

5.2.2.2. Decisión del tipo y formato del instrumento y su contexto de administración [Paso 2]

Se utilizará el procedimiento de recolección de datos mixto conformado por el cuestionario CDLP y luego por una entrevista. Tanto las preguntas como los ejercicios se entregan a los participantes en forma de cuestionario. Con el propósito de minimizar los errores de caracterización, una vez que el participante ha contestado el cuestionario se desarrollará una entrevista individual donde el entrevistador realiza preguntas a efectos de que el participante justifique sus respuestas y ante la correcta justificación del participante el entrevistador dará misma como válida.

Respecto al contexto de administración será una sala provista de una computadora por programador ya que la primer instancia, la que refiere al cuestionario, es auto administrada permitiendo llevar esta acción de forma individual o grupal, para luego si pasar a la instancia de entrevista de forma individual.

5.2.2.3. Construcción de la primera versión del Cuestionario [Paso 3]

Algunas cuestiones referidas al procedimiento serán explicadas por el investigador responsable, como por ejemplo: el tiempo aproximado de respuesta, el procesamiento de los cuestionarios, la confidencialidad del manejo de la información, etc.

5.2.2.3.1. Cuestionario de Caracterización Dependiente del Lenguaje de Programación C

A continuación se enumeran los ítems que forman parte del cuestionario que permite relevar las variables de caracterización que son dependientes del lenguaje de programación C.

1. ¿Cuántos años de experiencia tiene con el lenguaje de programación elegido?
    - Ninguno
    - Uno
    - Dos
    - Tres
    - Cuatro
    - Entre cinco y diez
    - Más de diez
2. ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional?
    - Ninguno
    - Uno
    - Dos
    - Tres
    - Cuatro
    - Entre cinco y diez
    - Más de diez

3. Relacione el nombre de la función con la descripción de la misma.

   scanf ___ strcmp ___ realloc ___ sizeof ___

   1. Lee exclusivamente strings de la entrada estándar.
   2. Compara dos cadenas de caracteres.
   3. Redimensiona memoria previamente reservada con malloc.
   4. Retorna la longitud de un array.
   5. Copia una cadena de caracteres en otra.
   6. Lee datos formateados de la entrada estándar.
   7. Reserva memoria de manera dinámica.
   8. Retorna el tamaño en bytes ocupado por un tipo de dato.

4. Relacione la palabra reservada con la descripción de su propósito.

   define ___ continue ___ typedef ___

   1. Sustituye la cadena de token por cada aparición del identificador.
   2. Causa la inmediata salida de una estructura de control.
   3. Da un nombre nuevo a cualquier tipo de datos.
   4. Declara una variable indicando que su valor es inalterable.
   5. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
   6. Define un nuevo tipo de dato.

5. ¿Qué es y para qué se utiliza una estructura?

6. ¿Qué es un puntero?

7. ¿En una función a qué se denomina pasaje por valor y pasaje por referencia?

8. ¿Si la variable op tiene valor ‘H’ cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case 'H' : printf("Hola ");
   case 'J' : printf("Juan ");
   case 'C' : printf("Chau ");
   break;
}
```

(A) Hola
(B) Hola Juan
(C) Hola Juan Chau
(D) Ninguna de las anteriores

9. ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
#include <stdio.h>
int main()
{
  int data[5], i;
  for(i = 5;  i > 0; i--)
    *(data + i)  = i;
  for(i = 0;  i < 5; i++)
     printf("%d-", *(data + i));
  return 0;
}
```

(A) 1-2-3-4-5-
(B) 5-4-3-2-1-
(C) 0-1-2-3-4-
(A) 4-3-2-1-0-
(D) Ninguna de las anteriores

10. ¿Cuál es el propósito del siguiente bloque de código?

```
int num=50, i=1;
while( i<=num){
        if( !( num % i)){
                printf(“%d”, i);
        }
i++;
}
```

a. Muestra números pares desde el 1 al 50
b. Muestra los divisores de 50
c. Muestra números impares desde el 1 al 50
d. Muestra los números que no son divisores de 50
e. Error de compilación
f. No muestra nada

11. A partir del siguiente código indique qué valor se muestra por la consola

```
int *p, *q;
int x = 4;
p = &x;
q = p;
*q += 3;
printf(“%d”, *p);
```

a. Muestra valores no determinados
b. Muestra el número 12
c. Muestra el número 4
d. Muestra la dirección de x
e. Error de compilación
f. Muestra el número 7

5.2.2.3.2. Cuestionario de Caracterización Dependiente del Lenguaje de Programación C#

A continuación se enumeran los ítems que forman parte del cuestionario que permite relevar las variables de caracterización que son dependientes del lenguaje de programación C#.

a) ¿Cuántos años de experiencia tiene con el Lenguaje C#?

- Ninguno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

b) ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional?

- Ninguno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

c) Relacione el nombre de los métodos con la descripción de los mismos.

Console ___ Object.Equals ___ Int32.TryParse ___ List.Count ___

1. Compara si dos objetos hacen referencia a una misma instancia
2. Retorna la cantidad de ítems de una lista
3. Intenta convertir un dato a entero. Si lo logra retorna el número convertido.
4. Compara dos objetos por su valor
5. Representa los flujos de salida para las aplicaciones de consola.
6. Intenta convertir un dato a entero. Si lo logra retorna true caso contrario retorna false.
7. Retorna el tamaño en bytes ocupado por la lista
8. Representa los flujos de entrada, salida y error estándar para las aplicaciones de consola.

d) Relacione la palabra reservada con la descripción de su propósito.

using ___ break ___ : (Ej. C1:C2) ___

1. Importa una clase.
2. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
3. Define una relación de herencia, donde C1 hereda de C2.
4. Importar espacios de nombres (namespaces).
5. Define una relación de herencia, donde C2 hereda de C1.
6. Causa la inmediata salida de una estructura de control.

e) ¿Qué diferencia existe entre un método abstract y uno virtual?

f) ¿Qué se puede definir dentro de una interfaz?

g) Si definó class A : B, C. ¿Es esto correcto? De serlo, ¿De qué tipo deberían ser B y C?

h) ¿Si la variable op tiene valor "H" cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case "H": Console.WriteLine("Hola ");
   case "J": Console.WriteLine("Juan ");
   case "C": Console.WriteLine("Chau ");
   break;
}
```

(A) Hola
(B) Hola Juan
(C) Hola Juan Chau
(D) Ninguna de las anteriores

i) ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
static int main(){
        int max = 5;
        int[] data = new int[max];
        int i;
        for (i = 5; i > 0; i--)
                data[max-i] = i;
        for (i = 0; i < 5; i++)
                Console.WriteLine("{0}-", data[i]);
        return 0;
}
```

(A) 1-2-3-4-5-
(B) 5-4-3-2-1-
(C) 0-1-2-3-4-
(A) 4-3-2-1-0-
(D) Ninguna de las anteriores

j) ¿Cuál es el propósito del siguiente bloque de código?

```
int num = 50, j = 1;
while( j <= num){
        if( (num % j) == 0 ){
                Console.WriteLine("{0}-", j);
        }
        j++;
}
```

(A) Muestra números pares desde el 1 al 50
(B) Muestra los divisores de 50
(C) Muestra números impares desde el 1 al 50
(D) Muestra los números que no son divisores de 50
(E) Error de compilación
(F) No muestra nada

k) A partir del siguiente código indique qué valor se muestra por la consola

```
Clase1 c1 = new Clase1 (12);
Clase1 c2 = c1;
c1.Dato = 445;
c2.Dato = 4;
Console.WriteLine(c1.Dato);
```

(A) Muestra valores no determinados
(B) Muestra el número 445
(C) Muestra el número 12
(D) Muestra la dirección de c1
(E) Error de compilación
(F) Muestra el número 4

5.3. Mecanismo de Formación de Pares Experimentales

Este procedimiento es de suma importancia ya que implica determinar el mecanismo mediante el cual serán elegidas las parejas de sujetos a ser comparados. Es pertinente contar con un mecanismo que asegure que las parejas experimentales se encuentran integradas por sujetos homogéneos, esto implica que según su caracterización son indistinguibles o presentan diferencias mínimas. A continuación se describe el criterio de utilización de las variables (Sección 5.3.1) las cuales surgen como resultado del proceso de caracterización, el proceso de normalización de las variables (5.3.2), el procedimiento de aplicación de las penalidades referidas al tiempo (Sección 5.3.3), el procedimiento de cálculo de distancia (Sección 5.3.4) y por último el algoritmo que se utilizará para elegir los pares experimentales (Sección 5.3.5).

5.3.1. Criterio de Utilización de las Variables

Del procedimiento de caracterización surgen múltiples variables, algunas relacionadas a características independientes del lenguaje de programación y otras a características que dependen del desempeño del programador en un lenguaje de programación en particular. Según las características del estudio puede o no tener sentido caracterizar al programador de manera dependiente al lenguaje utilizado, en el caso de esos estudios donde esto tenga sentido, es importante resaltar que la variable que refiere al lenguaje elegido por el programador deberá ser excluida tanto del proceso de normalización (Sección 5.2.2.) como del procedimiento de cálculo de distancia (Sección 5.2.3).

5.3.2. Normalización

El proceso de normalización implica transformar los valores de las variables independientes a fin de que estas queden representadas en el rango [0-10] independientemente de cual fuese su rango original. Este paso permitirá asegurar que ninguna de las variables intervinientes en el cálculo de distancia se encuentran ponderadas por sobre otras. Para llevar este procedimiento adelante se optó por utilizar una transformación lineal (Figura 5.4).

$$Vt = \frac{V}{Rmax - Rmin} \times (Tmax - Tmin)$$

Figura 5.4. Transformación lineal

Siendo 'V' el valor original de la variable, *'Rmax'* el valor máximo que puede adoptar la variable original, *'Rmin'* el valor mínimo que puede adoptar la variable original, *'Tmax'* el valor máximo que puede adoptar la variable luego de transformada y *'Rmin'* el valor mínimo que puede adoptar la variable luego de transformada, entonces, *'Vt'* representa el valor transformado de la variable.

5.3.3. Penalidades por tiempo

Cada una de las variables medida es acompañada por el tiempo insumido por el participante en completar los ejercicios relacionados a la misma. A efectos de formar pares experimentales que no solo sean indistinguibles en lo que refiere a conocimiento sino también al tiempo insumido para dar dicha respuesta se aplicará una penalidad en el puntaje obtenido según el tiempo insumido (Tabla 5.15).

Minutos		% del Puntaje
Desde	**Hasta**	
0	5	100%
5	10	90%
10	15	80%
15	20	70%
20	25	60%
25	30	50%
30	35	40%
35	40	30%
40	45	20%
45	50	10%
50	55	0%

Tabla 5.15. Penalidades por tiempo

5.3.4. Cálculo de Distancia

En un escenario donde se requiere evaluar múltiples variables, todas ellas cuantitativas y cuyos valores pertenecen luego del proceso de normalización al intervalo [0,10] se debe definir el criterio a ser empleado para determinar la distancia existente entre dos sujetos, por tratarse de un espacio n-dimensional se aplicará el cálculo de distancia euclídea (Figura 5.5).

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

Figura 5.5. Distancia euclídea

Por lo tanto la utilización de múltiples variables de caracterización no implica dificultad alguna a efectos de calcular la distancia euclídea de cada sujeto del grupo A con cada sujeto del grupo B. Con el propósito de mejorar la calidad del apareamiento se incorporará un restricción (Figura 5.6.), solo se realizará el cálculo de distancia entre participantes donde el valor del módulo de la diferencia entre variables del mismo tipo no supere un umbral denominado β .

$$|p_n - q_n| \leq \beta$$

Figura 5.6. Restricción

Esta restricción permitirá establecer una distancia máxima tolerada la cual no podrá ser producto de una diferencia significativa en una sola variable. El cálculo de la distancia euclídea dará como resultado una matriz (Tabla 5.16) en la cual se representan las distancias entre los participantes del Grupo A y los participantes del Grupo B que han cumplido con la restricción ($|p_n - q_n| \leq \beta$).

		GRUPO A				
		SA1	SA2	SA3	…	SAn
GRUPO B	SB1	0	3	1	.	1
	SB2	5	-	0	.	1
	SB3	1	0	3	.	1
	…	.	.	.	.	4
	SBn	1	1	2	3	4

Tabla5.16. Matriz de distancia

5.3.5. Algoritmo de Selección de Parejas Experimentales

El algoritmo de selección de parejas experimentales descripto en el (Anexo G) será aplicado a la matriz de distancia y como resultado se obtiene una tabla en la cual figura cada integrante del grupo A apareado con un integrante del grupo B y la distancia euclídea que existe entre ellos.

Sujeto Grupo A	Sujeto Grupo B	Distancia
SA1	SB1	0
SA2	SB3	0
SA3	SB2	0
…	…	…
SAn	SBn	4

Tablas 5.17. Distancias entre los pares experimentales

Una vez que se han formado los pares experimentales con dicho algoritmo solo resta determinar cuál es el valor máximo tolerado de distancia que permita asegurar que dos sujetos son indistinguibles y una vez definido dicho valor se procederá a eliminar todas las parejas que lo excedan.

5.4. Validación del Protocolo

En esta sección, se desarrollará el paso 4 (prueba piloto de los instrumentos de medición) y el paso 5 (Obtención de una versión mejorada del instrumento). Esta prueba piloto inicial tiene varios propósitos, someter a prueba los instrumentos de medición (Cuestionario CILP y los Cuestionarios CDLP) y evaluar las condiciones de aplicación y los procedimientos involucrados. También permitirá analizar si las instrucciones se comprenden por los usuarios, si los ítems funcionan de manera adecuada. Esta prueba piloto permitirá evaluar el lenguaje y la redacción de los instrumentos. Luego se analizarán los resultados obtenidos de la prueba piloto inicial y en función de ellos se determinará la necesidad de realizar cambios en las herramientas de medición.

5.4.1. Prueba piloto [Paso 4]

Con el propósito de probar los requisitos esenciales de los instrumentos de medición: confiabilidad, validez y objetividad, se decidió realizar una serie de acciones para demostrar su nivel de confiabilidad y validez inicial. Los instrumentos de caracterización antes descriptos tienen por propósito medir un conjunto de características de los programadores, algunas de las cuales serán tenidas en cuenta al momento de realizar la formación de los pares experimentales:

- Comprensión de una Especificación (CILP3)
- Comprensión de Pseudocódigo (CILP4)
- Capacidad Algorítmica (CILP5)
- Conocimiento Teórico (CDLP2)
- Comprensión de Código Fuente (CDLP3)

Y otras que han sido incluidas a solo efecto de demostrar que no guardan ninguna relación con el desempeño de los participantes, como ser:

- Nivel de formación (CILP1)
- Experiencia (CILP2)
- Experiencia con el Lenguaje Elegido (CDLP1)

Una vez medidas dichas características se buscará formar pares experimentales de programadores que utilizan el mismo lenguaje de programación a efecto de poder medir si dicha pareja experimental presenta o no diferencias significativas en la resolución de una misma tarea. Si los pares experimentales conformados aplicando el protocolo no presentan diferencias significativas al resolver con el mismo lenguaje la misma tarea, se podrá considerar que tanto los instrumentos como el protocolo de formación de pares experimentales se encuentran en una versión estable.

5.4.1.1. Escenario de desarrollo de la Prueba Piloto Inicial

Para la realización de la prueba piloto inicial, se convocó a participar de la investigación a personas mayores de edad que han manifestado conocer alguno de los lenguajes de programación que forman parte de la misma, logrando conformar un grupo de treinta programadores. Respetando los principios éticos planteados en el (Capítulo 4) se llevó adelante la elaboración del documento de consentimiento libre e informado (Anexo A), el cual será entregado a cada interesado a efectos de que pueda leerlo, analizarlo y en el caso de prestar su conformidad comenzar a formar parte de la investigación.

5.4.1.2. Procedimiento de la prueba piloto

En la (Figura 5.7) se describen las actividades a ser llevadas a cabo en la prueba piloto del protocolo de formación de pares experimentales de programadores, detallando no solo la actividad sino también cuáles son sus entradas y que se consigue como salida una vez concluida.

5.4.1.2.1. Caracterización independiente del objeto de estudio

Esta actividad tiene por propósito caracterizar a los programadores de manera independiente al lenguaje de programación para lo cual utiliza como insumo el instrumento de caracterización independiente del lenguaje de programación (CILP). Es importante en esta etapa luego de ser distribuidos los cuestionarios dedicar el tiempo necesario para contestar cualquier consulta que pudiera surgir relacionada con la comprensión de alguna de las preguntas del cuestionario. Se deberá tomar nota de todas las preguntas realizadas por los participantes a efectos de evaluar mejoras de la herramienta. Esta etapa arroja como resultado un CILP completado por cada uno de los participantes.

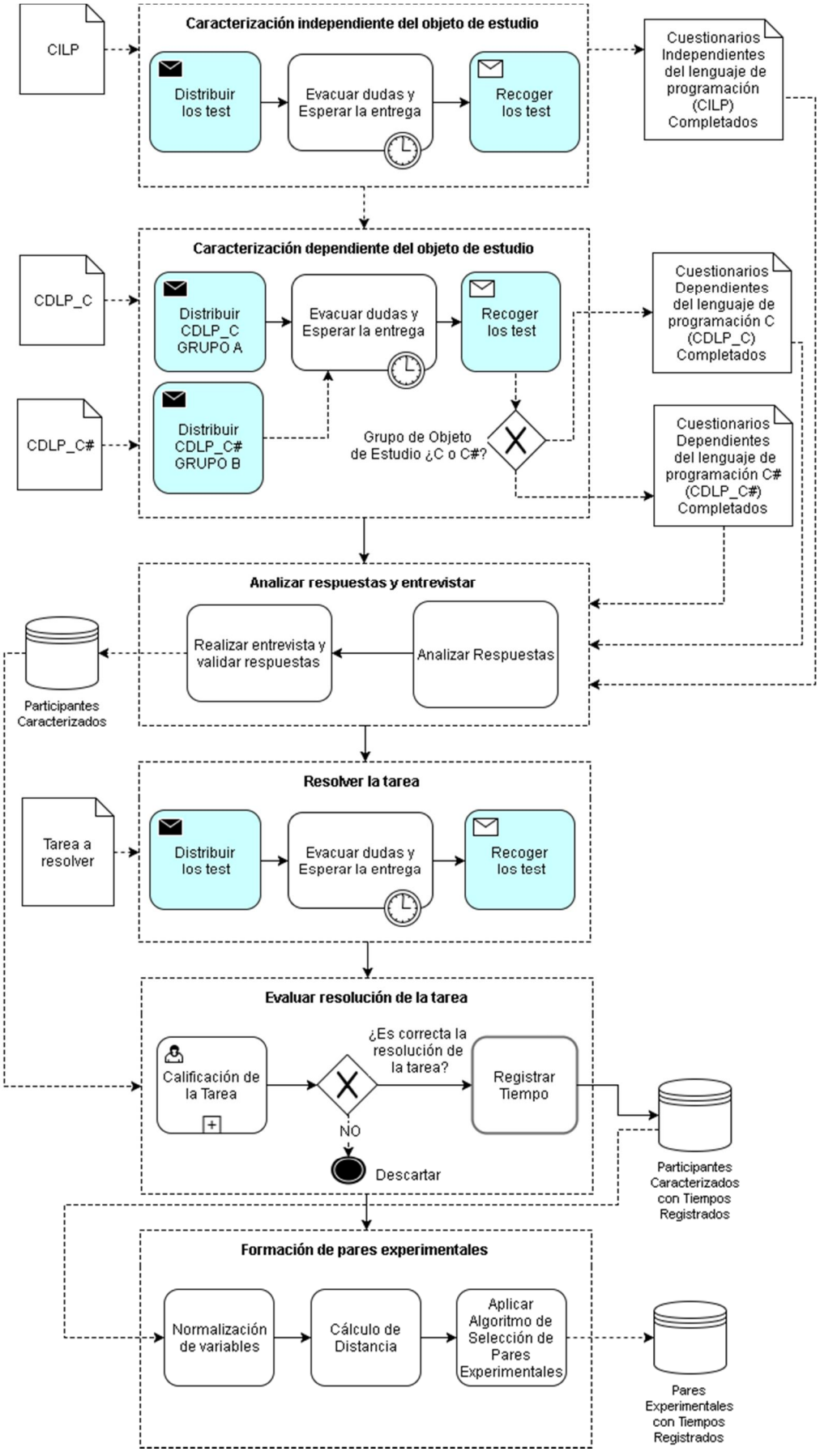

Figura 5.7. Procedimiento de la prueba piloto.

5.4.1.2.2. Caracterización dependiente del objeto de estudio

El propósito de esta actividad es caracterizar a los participantes teniendo en cuenta el lenguaje de programación que manifiestan conocer, por lo cual, algunos de los participantes utilizarán el instrumento desarrollado para caracterizar a los programadores del lenguaje C (CDLP_C) y otros el instrumento desarrollado para caracterizar a los programadores del lenguaje C# (CDLP_C#). Al igual que en la actividad anterior se deberá prestar especial atención a las preguntar realizadas por los participantes. Esta etapa da como resultado un CDLP completado por cada uno de los participantes, según el lenguaje elegido este será un CDLP_C o un CDLP_C#.

5.4.1.2.3. Analizar respuestas y realizar entrevistas

Luego de realizadas ambas caracterizaciones y analizadas las respuestas de cada participante se realiza una entrevista individual a efectos de validar las respuestas a las preguntas que refieren a la experiencia y al nivel de formación. Esta etapa da como resultado una nómina de participantes caracterizados.

5.4.1.2.4. Resolver la tarea

Se les solicitará a los participantes que realicen un programa que dé cumplimiento al mismo requerimiento de baja complejidad. Independientemente del lenguaje que cada participante utilice el requerimiento será el mismo. En esta etapa se determina si los participantes ya caracterizados se encuentran en condiciones de desarrollar la tarea y en tal caso cual es el tiempo que les demanda. Dando como resultado una propuesta de solución a la tarea planteada por cada participante junto al tiempo insumido en desarrollarla.

5.4.1.2.5. Evaluar resolución de la tarea

Se evaluará que la tarea cumpla con el requerimiento y sea funcional. Esta etapa da como resultado una nómina de participantes caracterizados que han podido desarrollar la tarea de manera correcta y en cada caso se registrará el tiempo insumido.

5.4.1.2.6. Formación de pares experimentales

Partiendo de la nómina de participantes caracterizados con el tiempo insumido en desarrollar la tarea, se comienza por normalizar las variables, esto busca que ninguna de las dimensiones este ponderada por sobre las otras al momento de realizar el cálculo de distancia. Luego se calculan las matrices de distancia que existe entre todos los participantes, obteniendo una para cada lenguaje de programación empleado. Por último se aplica el algoritmo de selección de pares experimentales, el

cual permite obtener los pares experimentales junto al tiempo empleado por cada uno de los integrantes de cada par.

5.4.1.2. Resultados de la Prueba Piloto Inicial

El procesamiento de los datos obtenidos se ha realizado en MS-Excel. Las reglas utilizadas para resolver los cálculos de la herramienta de medición son las siguientes:

- **Regla de Resolución 1**. La pregunta referida al *nivel de estudios del que posee título oficial. (CILP1.1)* que se encuentra en el Cuestionario CILP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 2**. La pregunta *¿Cuántas materias o cursos de programación ha realizado hasta el momento? (CILP1.2)* que se encuentra en el Cuestionario CILP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 3**. La pregunta *¿Cuántos años de experiencia tiene como programador? (CILP2.1)* que se encuentra en el Cuestionario CILP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 4**. La pregunta *¿En cuántos lenguajes de programación considera tener un nivel regular o superior? (CILP2.2)* que se encuentra en el Cuestionario CILP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 5**. La pregunta *¿Cuántos años de experiencia tiene con el lenguaje de programación elegido? (CDLP1.2)* que se encuentra en el Cuestionario CDLP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 6**. La pregunta *¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional? (CDLP1.3)* que se encuentra en el Cuestionario CDLP, no se contempló en los cálculos. Se realizó a solo efecto de demostrar que no guarda relación alguna con el desempeño de los participantes.
- **Regla de Resolución 7.** La dimensión *Comprensión de una Especificación. (CILP3),* se resuelve con la escala discreta de 0 a 4, aportando cada una de las cuatro preguntas un punto.

- **Regla de Resolución 8.** La dimensión *Comprensión de Pseudocódigo. (CILP4),* se resuelve con la escala discreta de 0 a 2, aportando cada una de las dos preguntas un punto.
- **Regla de Resolución 9.** La dimensión *Capacidad Algorítmica. (CILP5),* se resuelve promediando *CILP5.1 (*de escala discreta de 0 a 11) y *CILP5.2 (*de escala discreta de 0 a 3), el promedio realiza luego de normalizar ambas variables.
- **Regla de Resolución 10.** La dimensión *Conocimiento Teórico del Lenguaje (CDLP2),* tanto en el caso del lenguaje C como en el caso del lenguaje C#, se resuelve con la escala discreta de 0 a 10, aportando cada una de las diez preguntas un punto.
- **Regla de Resolución 11.** La dimensión *Comprensión de Código Fuente del Lenguaje. (CDLP3),* tanto en el caso del lenguaje C como en el caso del lenguaje C#, se resuelve con la escala discreta de 0 a 4, aportando cada una de las cuatro preguntas un punto.

A continuación se presentan los resultados de la caracterización dependiente del lenguaje de programación. La Tabla 5.18, contiene los resultados normalizados de la caracterización dependiente del lenguaje C y la Tabla 5.19, contiene los resultados normalizados de la caracterización dependiente del lenguaje C#; cabe la pena destacar que a pesar de tratarse de un grupo de programadores homogéneo en lo que refiere al lenguaje de programación que manifiestan conocer, los años de experiencia con el mismo y los años de uso profesional, dicha homogeneidad no se ve manifiesta en lo que refiere a conocimiento teórico y comprensión de código.

Sujeto	Lenguaje	Años de experiencia	Años de uso profesional	Conocimiento teórico	Comprensión de código
	CDLP1.1	CDLP1.2	CDLP1.3	CDLP2	CDLP3
1	C	0	0	2.4	0
2	C	0	0	8.1	6.75
3	C	0	0	6.4	4
4	C	0	0	9	9
5	C	0	0	6.3	2.25
6	C	0	0	7.2	6
7	C	0	0	8.1	0
8	C	0	0	5.6	2.25
9	C	0	0	8	4.5
10	C	0	0	6.3	4.5
11	C	0	0	7.2	2.5
12	C	0	0	9	6.75
13	C	0	0	5.6	0
14	C	2	0	6.4	0

Tabla 5.18. Resultados normalizados de la caracterización dependiente del lenguaje C

Sujeto	Lenguaje	Años de experiencia	Años de uso profesional	Conocimiento teórico	Comprensión de código
	CDLP1.1	CDLP1.2	CDLP1.3	CDLP2	CDLP3
15	C#	0	0	9	9
16	C#	0	0	4	5
17	C#	0	0	6.3	2.5
18	C#	2	1	9	2.25
19	C#	0	0	3.6	2.25
20	C#	0	0	7.2	6.75
21	C#	0	0	8	2.25
22	C#	0	0	5.4	6.75
44	C#	0	0	8.1	2.5
45	C#	0	0	7	2.5
46	C#	0	0	6	5
47	C#	0	0	8	5
48	C#	0	0	5	5
49	C#	0	0	8	5

Tabla 5.19. Resultados normalizados de la caracterización dependiente del lenguaje C#

Surge de analizar la (Tabla 5.18) la cual contiene los resultados normalizados de la caracterización independiente del lenguaje de programación de aquellos participantes que optaron por utilizar el lenguaje C y la (Tabla 5.19) que contiene igual información pero referida a los programadores que optaron por utilizar C# que a pesar de tratarse de un grupo de programadores homogéneos en lo que refiere a:

- Nivel de formación (CILP1)
- Experiencia (CILP2)
- Experiencia con el Lenguaje Elegido (CDLP1)

Dicha homogeneidad no se ve manifiesta en lo que refiere a conocimiento teórico y comprensión de código. Por lo cual tanto en la tabla (Tabla 5.20) la cual contiene los resultado normalizados de la caracterización junto al tiempo insumido de los programadores de C como en la tabla (Tabla 5.21) la cual contiene los resultado normalizados de la caracterización junto al tiempo insumido de los programadores de C#, se representan solo las dimensiones que serán tenidas en cuenta a la hora de formar los pares experimentales. En aquellos casos donde no figura el tiempo insumido por el programador para realizar la tarea se debe a que la propuesta de solución planteada satisface lo planteado como requerimiento. Por lo cual se genera tanto la tabla (Tabla 5.22) y la (Tabla 5.23)

que contienen los resultados normalizados de la caracterización de los programadores que completaron la tarea.

ID SUJETO	Nivel de formación	Cursos realizados	Años de experiencia	Cantidad de lenguajes	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica
	CILP1.1	CILP1.2	CILP2.1	CILP2.2	CILP3	CILP4	CILP5
1	2	1	2	2	10.00	0.00	1.64
2	5	1	0	2	9.00	5.00	6.36
3	1	2	0	1	7.50	5.00	2.73
4	5	2	0	1	4.50	5.00	6.36
5	1	1	0	1	6.75	7.00	2.18
6	1	1	0	1	4.50	9.00	0.67
7	1	5	0	0	7.50	4.50	1.48
8	1	2	0	1	10.00	4.50	4.43
9	1	3	2	1	6.75	9.00	2.96
10	1	0	0	3	4.50	4.50	7.64
11	5	4	0	1	6.75	5.00	1.86
12	1	3	0	1	9.00	9.00	5.45
13	1	0	0	0	7.50	5.00	7.00
14	2	3	1	1	9.00	4.50	3.87

Tabla 5.20. Resultados normalizados de la caracterización independiente del lenguaje sobre programadores de C

ID SUJETO	Nivel de formación	Cursos realizados	Años de experiencia	Cantidad de lenguajes	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica
	CILP1.1	CILP1.2	CILP2.1	CILP2.2	CILP3	CILP4	CILP5
15	2	1	0	4	10.00	4.50	5.91
16	1	2	1	1	10.00	0.00	2.86
17	1	5	1	1	7.50	10.00	7.00
18	2	4	2	2	6.75	5.00	5.18
19	2	3	0	1	5.00	5.00	
20	2	5	0	2	10.00	4.50	5.45
21	1	1	0	0	7.50	5.00	4.77
22	1	2	0	1	10.00	5.00	7.27
44	1	2	0	1	10.00	4.50	4.43
45	1	1	0	0	7.50	5.00	4.77
46	1	3	0	2	7.5	0	5.73
47	1	2	0	2	10	0	5.17
48	1	2	0	1	7.50	5.00	2.73
49	2	4	0	4	10	5	7.64

Tabla 5.21. Resultados normalizados de la caracterización independiente del lenguaje sobre programadores de C#

Sujeto	CDLP1.1	CDLP2	CDLP3	CILP3	CILP4	CILP5	Tiempo
1	C	2.4	0	10.00	0.00	1.64	-
2	C	8.1	6.75	9.00	5.00	6.36	79
3	C	6.4	4	7.50	5.00	2.73	-
4	C	9	9	4.50	5.00	6.36	85
5	C	6.3	2.25	6.75	7.00	2.18	121
6	C	7.2	6	4.50	9.00	0.67	-
7	C	8.1	0	7.50	4.50	1.48	105
8	C	5.6	2.25	10.00	4.50	4.43	-
9	C	8	4.5	6.75	9.00	2.96	110
10	C	6.3	4.5	4.50	4.50	7.64	-
11	C	7.2	2.5	6.75	5.00	1.86	118
12	C	9	6.75	9.00	9.00	5.45	115
13	C	5.6	0	7.50	5.00	7.00	-
14	C	6.4	0	9.00	4.50	3.87	100

Tabla 5.22. Resultados normalizados de la caracterización junto al tiempo insumido de los programadores de C

Sujeto	CDLP1.1	CDLP2	CDLP3	CILP3	CILP4	CILP5	Tiempo
15	C#	9	9	10.00	4.50	5.91	38
16	C#	4	5	10.00	0.00	2.86	-
17	C#	6.3	2.5	7.50	10.00	7.00	-
18	C#	9	2.25	6.75	5.00	5.18	60
19	C#	3.6	2.25	5.00	5.00	2.86	-
20	C#	7.2	6.75	10.00	4.50	5.45	33
21	C#	8	2.25	7.50	5.00	4.77	-
22	C#	5.4	6.75	10.00	5.00	7.27	39
44	C#	8.1	2.5	10.00	4.50	4.43	-
45	C#	7	2.5	7.50	5.00	4.77	65
46	C#	6	5	7.5	0	5.73	50
47	C#	8	5	10	0	5.17	58
48	C#	5	5	7.50	5.00	2.73	-
49	C#	8	5	10	5	7.64	41

Tabla 5.23. Resultados normalizados de la caracterización junto al tiempo insumido de los programadores de C #

ID	CDLP2	CDLP3	CILP3	CILP4	CILP5
2	8.10	6.75	9.00	5.00	6.36
4	9.00	9.00	4.50	5.00	6.36
5	6.30	2.25	6.75	7.00	2.18
7	8.10	0.00	7.50	4.50	1.48
9	8.00	4.50	6.75	9.00	2.96
11	7.20	2.50	6.75	5.00	1.86
12	9.00	6.75	9.00	9.00	5.45
14	6.40	0.00	9.00	4.50	3.87

Tabla 5.24. Resultados normalizados de la caracterización de los programadores de C que completaron la tarea

ID	CDLP2	CDLP3	CILP3	CILP4	CILP5
15	9.00	9.00	10.00	4.50	5.91
18	9.00	2.25	6.75	5.00	5.18
20	7.20	6.75	10.00	4.50	5.45
22	5.40	6.75	10.00	5.00	7.27
45	7.00	2.50	7.50	5.00	4.77
46	6.00	5.00	7.50	0.00	5.73
47	8.00	5.00	10.00	0.00	5.17
49	8.00	5.00	10.00	5.00	7.64

Tabla 5.25. Resultados normalizados de la caracterización de los programadores de C# que completaron la tarea

A continuación se presenta la matriz de distancia correspondiente a los programadores de C (Tabla 5.26) y correspondiente a los programadores de C# (Tabla 5.27). En ambas tablas se encuentran intersecciones pintadas de negro las cuales representan sujetos que no deben ser tenidos en cuenta a la hora de formar los pares experimentales ya que en por lo menos una de sus dimensiones han presentado una distancia superior a cinco. Los pares experimentales que surgen de aplicar el algoritmo se encuentran resaltados en color verde.

ID	2	4	5	7	9	11	12	14
2	-	5.11	7.07		6.14	6.65	4.20	
4	5.11	-			7.34		6.49	
5	7.07		-	3.95	3.54	2.23	6.88	4.38
7			3.95	-	6.58	2.83		9.20
9	6.14	7.34	3.54	6.58	-	4.67	4.17	7.00
11	6.65		2.23	2.83	4.67	-	7.44	4.03
12	4.20	6.49	6.88		4.17	7.44	-	
14			4.38	9.20	7.00	4.03		-

Tabla 5.26. Matriz de distancia de los programadores de C

ID	15	18	20	22	45	46	47	49
15	-		2.92	4.49		7.18	6.15	4.50
18		-	5.86	6.94	2.19	6.51	6.64	5.01
20	2.92	5.86	-	2.61	5.01	5.57	4.90	2.95
22	4.49	6.94	2.61	-	5.76	6.09	6.27	6.27
45		2.19	5.01	5.76	-	5.76	6.22	4.66
46	7.18	6.51	5.57	6.09	5.76	-	3.25	6.24
47	6.15	6.64	4.90	6.27	6.22	3.25	-	5.58
49	4.50	5.01	2.95	6.27	4.66	6.24	5.58	-

Tabla 5.27. Matriz de distancia de los programadores de C#

Por último se presentan las diferencias de tiempo que existen entre los programadores que integran cada par experimental, en la (Tabla 5.28) se presentan las diferencias de tiempos de los programadores que utilizaron el lenguaje de programación C y en la (Tabla 5.29) se presentan la diferencias de tiempos de los programadores que utilizaron el lenguaje de programación C#.

ID Sujetos		Diferencia de Tiempo
2	4	6
5	11	3
7	14	5
9	12	5

Tabla 5.28. Diferencias de tiempo de los pares experimentales de los programadores de C

ID Sujetos		Diferencia de Tiempo
18	45	5
20	22	6
46	47	8
15	49	3

Tabla 5.29. Diferencias de tiempo de los pares experimentales de los programadores de C#

5.4.1.3. Análisis de los resultados de la prueba piloto inicial

En lo que refiere a la formación de los pares experimentales de programadores el protocolo ha funcionado correctamente y por lo tanto se considera que la herramienta tiene un nivel de confiabilidad aceptable. Puede observarse en la (Tabla 5.28) y en la (Tabla 5.29) que los pares experimentales de programadores presentan diferencias en el tiempo empleado para resolver la misma tarea con el mismo lenguaje de programación inferiores a los ocho minutos en todos los casos.

En lo referente al tiempo empleado para realizar la caracterización, el tiempo promedio utilizado para llevar adelante la caracterización independiente del lenguaje de programación fue de 28 minutos, para la caracterización dependiente del lenguaje de programación el tiempo promedio empleado fue de 19 minutos y para llevar a cabo las entrevistas en promedio se utilizaron 6 minutos por participante.

En la (Tabla 5.30) se presentan las observaciones de los participantes y las acciones a seguir en lo que refiere la prueba piloto inicial del cuestionario de caracterización independiente del lenguaje de programación (CILP) y en la (Tabla 5.31) se presentan las observaciones de los participantes y las acciones a seguir en lo que refiere la prueba piloto inicial del cuestionario de caracterización dependiente del lenguaje de programación (CDLP).

Observaciones de los participantes	Acciones a seguir
1. Algunos participantes al ser consultados sobre cuántas materias o cursos de programación han realizado en la pregunta (CILP1.2) consultaron sobre qué materias tener en cuenta y sobre si se podían o no contabilizar los cursos en línea.	1. Se les explicó que se refiere tanto a cursos como a materias en los cuales el objeto de estudio haya sido algoritmia o un lenguaje de programación en particular. Por otra parte se aclaró que tanto los cursos presenciales como los cursos realizados a distancia se podían contabilizar en tanto y en cuanto hayan sido completados. Se decidió dejar esto aclarado en la versión mejorada del instrumento.
2.- Algunos participantes en referencia a la pregunta (CILP2.1) consultaron si debían de contar solo la experiencia profesional.	2. Se les explicó que se refiere a los años que llevan programando ya sea de manera amateur o profesional. Se decidió dejar esto aclarado el en la versión mejorada del instrumento.

Tabla 5.30. Observaciones de los participantes y las acciones a seguir (Prueba piloto inicial del CILP).

Observaciones de los participantes	Acciones a seguir
1. En referencia a la pregunta (CDLP1.2) la cual refiere a los años de experiencia que el participante tiene con el lenguaje de programación elegido, surgieron preguntas relacionadas a si se debía contabilizar desde que se conoce el lenguaje hasta la actualidad o tratar de estimar el tiempo real de uso.	1. Se explicó que se debía contabilizar desde la fecha en que comenzaron a utilizar dicho lenguaje hasta la actualidad. Se decidió dejar esto aclarado el en la versión mejorada del instrumento.
2. En referencia a la pregunta (CDLP1.3) referida a los años que el participante utilizó el lenguaje de programación elegido de manera profesional se consultó que debe tomarse como uso profesional.	2. Se les explicó que se refiere al uso del lenguaje de programación fuera del ámbito académico, entendiendo por esto a la utilización de dicho lenguaje para dar solución a problemas reales. Igualmente se decidió aclara el término 'profesional' en la versión mejorada del instrumento.
3.- Algunos participantes expresaron no comprender el término *token* (CDLP1.3)	3. Se ha formulado nuevamente la pregunta con una terminología más coloquial.

Tabla 5.31. Observaciones de los participantes y las acciones a seguir (Prueba piloto inicial del CDLP).

5.4.1.4. Obtención de una versión mejorada de la herramienta de medición [Paso 5]

Con todas las observaciones planteadas y las acciones realizadas en el punto anterior, se propone una segunda versión de los *cuestionarios* ***CILP, CDLP_C*** *y* ***CDLP_C#****.*

Los ítems que forman parte de la versión mejorada del cuestionario (Anexo D) de caracterización independiente del lenguaje de programación (CILP), se presenta a continuación:

A. De las siguientes alternativas seleccione la que corresponde al nivel educativo máximo del que posee título oficial
   - Primaria
   - Secundaria no técnica
   - Secundaria técnica
   - Terciaria
   - Universitaria de pregrado
   - Universitaria de grado
   - Universitaria de posgrado

B. ¿Cuántas materias o cursos de programación ha realizado hasta el momento? La pregunta se refiere tanto a cursos como a materias en los cuales el objeto de estudio haya sido algoritmia o un lenguaje de programación en particular. Se pueden contabilizar tanto cursos presenciales como los cursos realizados a distancia en tanto y en cuanto hayan sido completados.
   - Ninguno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

C. ¿Cuántos años hace que programa? No es relevante si no ha realizado esta actividad de manera profesional.
   - Menos de uno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

D. Suponiendo la siguiente escala de nivel que un programador tiene en un lenguaje de programación (Muy Malo, Malo, Regular, Bueno y Muy Bueno) ¿En cuántos lenguajes de programación considera tener un nivel regular o superior?
   - Menos de uno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

E. Suponiendo que se desea escribir en pseudocódigo un algoritmo capaz de procesar un archivo de texto y en cada línea del mismo se encuentra un numero comprendido en el rango [0 - 99]. El algoritmo deberá recorrer el archivo a efectos de informar por pantalla en qué posición se encuentra el número más alto, en el caso de repetirse dicho número en más de una línea se deberá de informar la primera ocurrencia. La ruta a dicho archivo se le solicitará al usuario al inicio del programa.

   1. Indique el resultado que se espera obtener al ejecutar el algoritmo
      - La cantidad de ocurrencias del número más grande.
      - El valor del número más grande.
      - La posición de cada uno de los números con valor igual al número más grande.
      - La posición de la primera ocurrencia del número con valor igual al número más grande.
      - Ninguna de las anteriores

   2. Indique si existe la necesidad de solicitar algún dato al usuario y en tal caso cual
      - NO.
      - SI, se solicita la cantidad de líneas del archivo.
      - SI, se solicita el nombre del archivo.
      - SI, se solicita la ruta al archivo.

   3. Indique si existe la necesidad de utilizar una estructura condicional
      - NO.
      - SI.

   4. Indique si existe la necesidad de utilizar una estructura repetitiva
      - NO.
      - SI.

F. Suponiendo el siguiente bloque de pseudocódigo :

```
FOR X = 1 to 100
        IF X%10 == 0 THEN
                PRINT X;
                X=X*2;
        END IF
END FOR
```

¿Cuál es la salida por pantalla?

- Números que son múltiplo de diez entre uno y cien
- Números que no son múltiplo de diez entre uno y cien
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Ninguna de las anteriores.

G. Suponiendo el siguiente bloque de pseudocódigo :

```
NUM X=49;
NUM Z=0;
WHILE X < 50
        PRINT X;
        IF Z== 5 THEN
                X=X+1;
        END IF
        Z=Z+1;
END WHILE
```

¿Cuál es la salida por pantalla?

- Números entre cero y cuarenta y nueve.
- Solo el número cuarenta y nueve.
- El número cuarenta y nueve en cinco ocasiones
- El número cuarenta y nueve en seis ocasiones
- Ninguna de las anteriores.

H. Luego de ser completados los lugares en blanco de la (Figura 5.8) el programa deberá permitir validar los datos de acceso de un usuario al sistema. El usuario tendrá como nombre de acceso ‘admin’ y como contraseña ‘2357’. El programa deberá dar como máximo tres oportunidades. Si el usuario ingresa un nombre de cuenta inválido el mensaje a mostrar será ‘USUARIO NO REGISTRADO’. Si ingresa bien el nombre de usuario y la contraseña es incorrecta el mensaje será ‘CONTRASEÑA INVALIDA’. Si tanto el nombre como la contraseña son correctos el mensaje será ‘BIENVENIDO’. En el caso de exceder la cantidad de intentos el programa deberá finalizar.

```
1  NUM intento = 0;
2  STR usuario;
3   0  password;
4        1
5   2   (intento  3  3)
6  {
7       usuario =    4    ;
8       password = READ();
9
10      IF( usuario 5 "admin")
11      {
12          PRINT "USUARIO NO REGISTRADO";
13          .   6   ++;
14
15      }
16      7
17      {
18          IF( contraseña  8  "2357")
19          {
20              registrado = 1;
21              BREAK;
22          }
23          PRINT "CONTRASEÑA INVALIDA"
24          intento 9 ;
25      }
26 }
27 IF( registrado 10  1)
28 {
29      PRINT "BIENVENIDO"
30 }
```

Figura 5.8. Ejercicio a completar

I. Escriba en pseudocódigo un programa que calcule el sueldo neto de un trabajador quien cobra según las horas trabajadas. El cálculo se realiza de la siguiente forma:

- El valor de la hora es $100
- Las primeras 160 horas a una tarifa fija.
- Las horas extras se pagan a un 50% más de la tarifa fija.
- Los impuestos a deducir de los trabajadores varían según el sueldo mensual. Si el sueldo supera los $15000 el trabajador pagara un 10% de impuesto.

El programa deberá solicitar al usuario la cantidad de horas trabajadas e indicar por pantalla el sueldo neto que el trabajador percibirá por sus servicios.

A continuación se enumeran los ítems que forman parte del cuestionario (Anexo E) mejorado que permite relevar las variables de caracterización que son dependientes del lenguaje de programación C (**CDLP_C**).

a) ¿Cuántos años de experiencia tiene con el lenguaje de programación elegido? Se debe de contabilizar desde la fecha en que comenzó a utilizar el lenguaje hasta la actualidad.
   - Ninguno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

b) ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional? Se refiere al uso del lenguaje de programación fuera del ámbito académico, entendiendo por esto a la utilización de dicho lenguaje para dar solución a problemas reales.
   - Ninguno
   - Uno
   - Dos
   - Tres
   - Cuatro
   - Entre cinco y diez
   - Más de diez

c) Relacione el nombre de la función con la descripción de la misma.

   scanf ___ strcmp ___ realloc ___ sizeof ___

   1. Lee exclusivamente strings de la entrada estándar.
   2. Compara dos cadenas de caracteres.
   3. Redimensiona memoria previamente reservada con malloc.
   4. Retorna la longitud de un array.
   5. Copia una cadena de caracteres en otra.
   6. Lee datos formateados de la entrada estándar.
   7. Reserva memoria de manera dinámica.
   8. Retorna el tamaño en bytes ocupado por un tipo de dato.

d) Relacione la palabra reservada con la descripción de su propósito.

define ___ continue ___ typedef ___

1. Se utiliza para crear macros
2. Causa la inmediata salida de una estructura de control.
3. Da un nombre nuevo a cualquier tipo de datos.
4. Declara una variable indicando que su valor es inalterable.
5. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
6. Define un nuevo tipo de dato.

e) ¿Qué es y para qué se utiliza una estructura?

f) ¿Qué es un puntero?

g) ¿En una función a qué se denomina pasaje por valor y pasaje por referencia?

h) ¿Si la variable op tiene valor 'H' cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case 'H' : printf("Hola ");
   case 'J' : printf("Juan ");
   case 'C' : printf("Chau ");
   break;
}
```

(A) Hola
(B) Hola Juan
(C) Hola Juan Chau
(D) Ninguna de las anteriores

i) ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
#include <stdio.h>
int main()
{
  int data[5], i;
  for(i = 5;  i > 0; i--)
    *(data + i)  = i;
  for(i = 0;  i < 5; i++)
     printf("%d-", *(data + i));
  return 0;
}
```

(A) 1-2-3-4-5-
(B) 5-4-3-2-1-
(C) 0-1-2-3-4-
(A) 4-3-2-1-0-
(D) Ninguna de las anteriores

j) ¿Cuál es el propósito del siguiente bloque de código?

```
int num=50, i=1;
while( i<=num){
        if( !( num % i)){
                printf("%d", i);
        }
i++;
}
```

a. Muestra números pares desde el 1 al 50
b. Muestra los divisores de 50
c. Muestra números impares desde el 1 al 50
d. Muestra los números que no son divisores de 50
e. Error de compilación
f. No muestra nada

k) A partir del siguiente código indique qué valor se muestra por la consola

```
int *p, *q;
int x = 4;
p = &x;
q = p;
*q += 3;
printf("%d", *p);
```

a. Muestra valores no determinados
b. Muestra el número 12
c. Muestra el número 4
d. Muestra la dirección de x
e. Error de compilación
f. Muestra el número 7

A continuación se enumeran los ítems que forman parte del cuestionario mejorado (Anexo F) que permite relevar las variables de caracterización que son dependientes del lenguaje de programación C# (**CDLP_C#**):

a) ¿Cuántos años de experiencia tiene con el Lenguaje C#?

- Ninguno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

b) ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional?

- Ninguno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

c) Relacione el nombre de los métodos con la descripción de los mismos.

Console ___ Object.Equals ___ Int32.TryParse ___ List.Count ___

1. Compara si dos objetos hacen referencia a una misma instancia
2. Retorna la cantidad de ítems de una lista
3. Intenta convertir un dato a entero. Si lo logra retorna el número convertido.
4. Compara dos objetos por su valor
5. Representa los flujos de salida para las aplicaciones de consola.
6. Intenta convertir un dato a entero. Si lo logra retorna true caso contrario retorna false.
7. Retorna el tamaño en bytes ocupado por la lista
8. Representa los flujos de entrada, salida y error estándar para las aplicaciones de consola.

d) Relacione la palabra reservada con la descripción de su propósito.

using ___ break ___ : (Ej. C1:C2) ___

1. Importa una clase.
2. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
3. Define una relación de herencia, donde C1 hereda de C2.
4. Importar espacios de nombres (namespaces).
5. Define una relación de herencia, donde C2 hereda de C1.
6. Causa la inmediata salida de una estructura de control.

e) ¿Qué diferencia existe entre un método abstract y uno virtual?

f) ¿Qué se puede definir dentro de una interfaz?

g) Si se define class A : B, C. ¿Es esto correcto? De serlo, ¿De qué tipo deberían ser B y C?

h) ¿Si la variable op tiene valor "H" cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case "H": Console.WriteLine("Hola ");
   case "J": Console.WriteLine("Juan ");
   case "C": Console.WriteLine("Chau ");
   break;
}
```

(A) Hola
(B) Hola Juan
(C) Hola Juan Chau
(D) Ninguna de las anteriores

i) ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
static int main(){
        int max = 5;
        int[] data = new int[max];
        int i;
        for (i = 5; i > 0; i--)
                data[max-i] = i;
        for (i = 0; i < 5; i++)
                Console.WriteLine("{0}-", data[i]);
        return 0;
}
```

(A) 1-2-3-4-5-
(B) 5-4-3-2-1-
(C) 0-1-2-3-4-
(A) 4-3-2-1-0-
(D) Ninguna de las anteriores

j) ¿Cuál es el propósito del siguiente bloque de código?

```
int num = 50, j = 1;
while( j <= num){
        if( (num % j) == 0 ){
                Console.WriteLine("{0}-", j);
        }
        j++;
}
```

(A) Muestra números pares desde el 1 al 50
(B) Muestra los divisores de 50
(C) Muestra números impares desde el 1 al 50
(D) Muestra los números que no son divisores de 50
(E) Error de compilación
(F) No muestra nada

k) A partir del siguiente código indique qué valor se muestra por la consola

```
Clase1 c1 = new Clase1 (12);
Clase1 c2 = c1;
c1.Dato = 445;
c2.Dato = 4;
Console.WriteLine(c1.Dato);
```

(A) Muestra valores no determinados
(B) Muestra el número 445
(C) Muestra el número 12
(D) Muestra la dirección de c1
(E) Error de compilación
(F) Muestra el número 4

6. CASO DE APLICACIÓN

El propósito de este capítulo es llevar adelante un experimento que permita aplicar el protocolo de formación de pares experimentales de programadores. Se trata de un caso en el cual se busca determinar si un lenguaje de programación es más productivo que otro a la hora de resolver un tipo de tarea determinada. El protocolo de formación de pares experimentales permitirá conformar un conjunto de parejas integradas por un programador de cada lenguaje de programación a efecto de poder determinar si alguno de los lenguajes de programación evaluados incide en la productividad. Considerando que los integrantes de cada pareja son homogéneos en lo que refiere a sus conocimientos y habilidades. Se comenzará por definir el diseño experimental (Sección 6.1) para luego pasar a la operación del experimento (Sección 6.2).

6.1. Diseño Experimental

El procedimiento experimental se basa en lo propuesto por [Wohlin et al., 2012] en su libro "Experimentation in Software Engineering", allí se plantea dividir el procedimiento en seis actividades principales. Alcance (Sección 6.1.1), se define el experimento en términos de objetivos. La planificación (Sección 6.1.2), donde se determina el diseño del experimento, se considera la instrumentación y se evalúan las amenazas al mismo. La operación del experimento (Sección 6.1.3) en esta actividad se obtienen las mediciones que luego se analizan y evalúan. Finalmente, los resultados, análisis e interpretación (Sección 6.1.4), el resumen (Sección 6.1.5) y las conclusiones (Sección 6.1.6).

6.1.1. Alcance

El alcance del experimento se establece definiendo sus objetivos. El propósito de una correcta definición de objetivos es asegurar que los aspectos importantes de un experimento se encuentran claramente establecidos antes de que se produzca la planificación y la ejecución [Wohlin et al., 2012].

6.1.1.1. Objetivo

El objetivo de los estudios empíricos que apliquen este procedimiento será determinar sí existen diferencias de productividad significativas en la resolución de una tarea determinada según el lenguaje de programación utilizado para llevar dicha tarea a cabo.

6.1.1.1.1. Propósito

El propósito del experimento será evaluar el desempeño de los programadores con el fin de determinar si la utilización de un lenguaje en programación determinado afecta o no su productividad.

6.1.1.1.2. Objeto de Estudio

El objeto de estudio serán los programas desarrollados en alguno de los lenguajes de programación a ser comparados, teniendo en cuenta que dichos lenguajes son utilizados por los programadores para resolver una tarea determinada y por lo tanto se deberán arbitrar los medios necesarios para evitar que las diferencias existentes entre estos afecte a la medición.

6.1.1.1.3. Perspectiva

Desde el punto de vista del investigador se buscará determinar sí existen diferencias sistemáticas en el rendimiento de un programador según el lenguaje que éste utiliza.

6.1.1.1.4. Foco

El principal efecto estudiado en el experimento es el rendimiento del programador aplicando un lenguaje determinado. La elección es centrarse en la productividad entendiendo a ésta como el tiempo insumido en desarrollar una solución de software de una manera adecuada.

6.1.1.2. Resumen del alcance

Analizar: *los programas desarrollados en un lenguaje de programación en particular*
Con el propósito de: *evaluar*
Con respecto a: *si su elección afecta a la productividad de los programadores*
Desde el punto de vista: *del investigador*

6.1.2. Planificación

Como en todo tipo de actividades de ingeniería, el experimento debe planificarse y los planes deben seguirse para controlar el experimento. El resultado del experimento puede alterarse o incluso destruirse si no se planifica adecuadamente [Wohlin et al., 2012] .

6.1.2.1. Selección de contexto

El contexto del experimento hace referencia al lugar donde este se llevará a cabo y a que grupo de personas participaran del mismo, este punto deberá ser definido en cada caso donde se requiera aplicar el procedimiento. La capacidad de generalizar los resultados a partir de este contexto específico se expone más adelante cuando se discuten amenazas a la validez en el experimento.

6.1.2.2. Formulación de la Hipótesis

Se ha elegido centrarse en una hipótesis, la cual indica que independientemente del lenguaje de programación utilizado no existen diferencias significativas desde el punto de vista de la productividad (medida como tiempo de desarrollo total).

6.1.2.2.1. Hipótesis nula

No existen diferencias significativas en la productividad (medida como tiempo de desarrollo total) entre los desarrollos realizados en el lenguaje de programación A (LA) y el lenguaje programación B (LB).

H0: Prod (LA) = Prod (LB)

6.1.2.2.2. Hipótesis alternativas

Existen diferencias significativas en la productividad (medida como tiempo de desarrollo total) entre los desarrollos realizados en el lenguaje de programación A (LA) y el lenguaje programación B (LB).

H1: Prod (LA) > Prod (LB)

H2: Prod (LA) < Prod (LB)

6.1.2.2.3. Datos a ser recolectados

Las hipótesis indican que los siguientes datos deben ser recolectados:

- Lenguaje de Programación Elegido: Lenguaje A ó Lenguaje B (escala nominal).
- La productividad medida como tiempo total insumido para desarrollar la tarea.
- Caracterización del Programador Independiente del Lenguaje de Programación.
- Caracterización del Programador Dependiente del Lenguaje de Programación.

6.1.2.3. Selección de variables

Como parte del diseño se deben elegir tanto las variables independientes como la variable dependiente. Elegir las variables correctas no es fácil y por lo general requiere conocimientos de dominio. Las variables independientes deben tener algún efecto sobre la variable dependiente y deben ser controlables.

6.1.2.3.1. Variables Independientes

Las variables independientes son aquellas variables que podemos controlar y cambiar en el experimento. Lo que refiere a la elección de las variables independientes fue tratado en el (Capítulo 5) a continuación solo serán enumeradas:

- Caracterización Independiente del Lenguaje de Programación
  - Nivel de formación (CILP1)
  - Experiencia (CILP2)
  - Comprensión de una Especificación (CILP3)
  - Comprensión de Pseudocódigo (CILP4)
  - Capacidad Algorítmica (CILP5)
- Caracterización Dependiente del Lenguaje de Programación
  - Experiencia con el Lenguaje Elegido (CDLP1)
  - Conocimiento Teórico (CDLP2)
  - Comprensión de Código Fuente (CDLP3)

6.1.2.3.1. Variable Dependiente

La variable dependiente del estudio es la productividad [PROD] medida como tiempo total insumido para desarrollar la tarea de manera adecuada.

6.1.2.4. Selección de los participantes

En general los participantes no son elegidos de manera aleatoria, estos se eligen en función de las posibilidades de quien lleva adelante el experimento, razón por la cual se debe describir cómo se lleva adelante esta selección y los criterios empleados para formar el grupo experimental.

6.1.2.5. Diseño del experimento

Planteado el problema y elegidas las variables independientes y dependientes es posible diseñar el experimento. El primer paso es abordar los principios generales de diseño:

6.1.2.5.1. Aleatoriedad

La tarea que permitirá que los programadores elaboren el objeto de estudio no está asignada aleatoriamente a los sujetos. Todos los participantes deberán resolver la misma tarea independientemente del lenguaje de programación utilizado. El objetivo del estudio no es evaluar al mismo programador frente a distintas tareas. El estudio busca comparar a dos programadores que utilizan lenguajes de programación distintos y que según la caracterización son indistinguibles.

6.1.2.5.2. Agrupamiento

Los participantes del experimento, tienen experiencias y habilidades diferentes. Para minimizar el efecto de la experiencia, se agrupará a los sujetos en pares experimentales, es decir se formarán parejas integradas por un sujeto que utiliza el lenguaje A y otro que utiliza el lenguaje B y que según la caracterización propuesta en el Capítulo 5 son indistinguibles.

6.1.2.5.3. Balanceo

El experimento utiliza un diseño equilibrado, lo que significa que luego de aplicar el protocolo de formación de pares experimentales existe el mismo número de personas que han optado por resolver la tarea utilizando el lenguaje de programación A y el lenguaje de programación B.

6.1.2.5.4. Tipo de diseño estándar

Una vez aplicado el protocolo de formación de pares experimentales (Capitulo 5) y realizada la tarea es posible aplicar una prueba no paramétrica a efectos de validar o refutar la hipótesis nula. Por lo tanto, la prueba de rangos con signo de Wilcoxon [1945] es adecuada para la evaluación. Esta prueba no paramétrica se utiliza para comprobar que no hay diferencia significativa en lo que refiere a la productividad entre los programadores que utilizan el lenguaje de programación A y los que usan el lenguaje de programación B. La prueba de rangos con signo de Wilcoxon permite analizar la similitud entre un conjunto de datos muestrales apareados donde cada elemento de la población posee un valor experimental que se desea comprobar y un valor de referencia o de control [Guardía-Olmos et al., 2007]. Como resultado de la prueba, es posible aceptar o refutar una hipótesis nula (H0) la cual indica que la población de diferencias entre los valores experimentales y los valores de control tienen una mediana de cero [Triola, 2013].

6.1.2.6. Instrumentación

Tanto la experiencia como las habilidades de los programadores que forman parte del estudio son tenidas en cuenta en la aplicación del protocolo de formación de pares experimentales (Capítulo 5). Estos datos caracterizan a los participantes, y por lo tanto son variables independientes en el experimento junto al lenguaje de programación que el participante opta por utilizar. Los objetos de estudio son los programas desarrollados en cada uno de los lenguajes de programación. Se verificará que estos programas satisfagan lo solicitado como tarea y que lo hagan de una manera correcta.

6.1.2.7. Evaluación de la validez

Se detectan tres niveles de amenazas de validez a considerar; validez interna (Sección 6.1.2.7.1.), validez externa (Sección 6.1.2.7.2.) y validez de la conclusión (Sección 6.1.2.7.3.).

6.1.2.7.1.Validez interna

La validez interna dentro de la población de alumnos de la Universidad que se encuentran actualmente cursando la Tecnicatura Superior en Programación no es un problema ya que el número de muestras se considera significativo y garantiza la validez interna.

6.1.2.7.2.Validez Externa

Es muy probable que se obtengan resultados equivalentes cuando se ejecute el mismo experimento en otro grupo de sujetos con características similares. Si se varía la caracterización, la tarea o la característica de los sujetos no será posible generalizar los resultados.

6.1.2.7.3. Validez de la Conclusión

La principal amenaza en cuanto a la validez de la conclusión es la calidad de los datos recogidos ya sea al aplicar el protocolo de formación de pares experimentales como así también en ejecución de la tarea. Es pertinente asegurar que tanto la tarea como la caracterización de los programadores se desarrollen en un ambiente controlado, el cual asegure que todos los sujetos cuentan con las mismas posibilidades a la hora de resolver el problema planteado. Dependiendo de cómo es llevado adelante el experimento existe el riesgo de que los resultados no reflejen la realidad, por lo tanto es pertinente llevar a cabo el procedimiento de manera rigurosa. Esto implica que el experimento en su totalidad sea supervisado por expertos, asegurando de esta manera que la validez de la conclusión no se verá afectada.

6.2. Operación del Experimento

El experimento busca comparar si el lenguaje de programación C presenta diferencias significativas en lo referido a productividad frente al lenguaje de programación C#. Para lo cual la operación del mismo se divide en tres etapas, en la primera de ellas se agrupan las tareas que guardan relación con la preparación del mismo (Sección 6.2.1.) y en la segunda etapa las tareas necesarias para la aplicación del protocolo de formación de pares experimentales (Sección 6.2.2.).

6.2.1. Preparación

Esta etapa se encuentra conformada por la convocatoria (Sección 6.2.1.1.) la firma del consentimiento libre e informado (Sección 6.2.1.2.) y la jornada de entrenamiento (Sección 6.2.1.3.).

6.2.1.1. Convocatoria

Se convocó a participar de la investigación a alumnos del primer año de la carrera de Técnico Superior en Programación de la Regional Avellaneda de la Universidad Tecnológica Nacional. Logrando conformar un grupo de sesenta participantes, todos ellos mayores de edad y que han manifestado conocer alguno de los lenguajes de programación intervinientes.

6.2.1.2. Consentimiento libre e informado

Respetando los principios éticos planteados en el (Capítulo 4) se llevó adelante la elaboración del documento de consentimiento libre e informado (Anexo A) el cual fue entregado a cada interesado a efectos de que pueda leerlo, analizarlo y en el caso de prestar su conformidad comenzar a formar parte de la investigación.

6.2.1.3. Entrenamiento

La jornada de entrenamiento tiene por propósito repasar los conocimientos necesarios para realizar la tarea y permitir que los participantes interactúen con el entorno de desarrollo donde se llevará adelante dicha tarea. El tiempo que se destine a esta jornada puede variar en cada experimento en función del grupo de sujetos elegido para formar parte del mismo. En el caso de esta experiencia la jornada de entrenamiento se acotó a un periodo de cuatro horas por tratarse en todos los casos de alumnos que ya dominaban el entorno de desarrollo.

6.2.2. Aplicación del protocolo de formación de pares experimentales

Esta etapa se encuentra conformada por las tareas propias de la aplicación del protocolo de formación de pares experimentales las cuales se encuentran descriptas en el (Capítulo 5), la (Figura 6.1) se describe el procedimiento que se llevará a cabo para su aplicación.

En la tabla (Tabla 6.1.) se presentan los resultado normalizados de la caracterización todos los programadores que optaron por utilizar el lenguaje de programación C. En la tabla (Tabla 6.2.) se presentan los resultado normalizados de la caracterización todos los programadores que optaron por utilizar el lenguaje de programación C#. En la tabla (Tabla 6.3.) se presentan los resultado normalizados de la caracterización junto al tiempo insumido en resolver la tarea del (Anexo C) de

aquellos programadores que utilizando el lenguaje C lo han conseguido satisfactoriamente. En la tabla (Tabla 6.4.) se presentan los resultados normalizados de la caracterización junto al tiempo insumido en resolver la tarea del (Anexo C) de aquellos programadores que utilizando el lenguaje C# lo han conseguido satisfactoriamente.

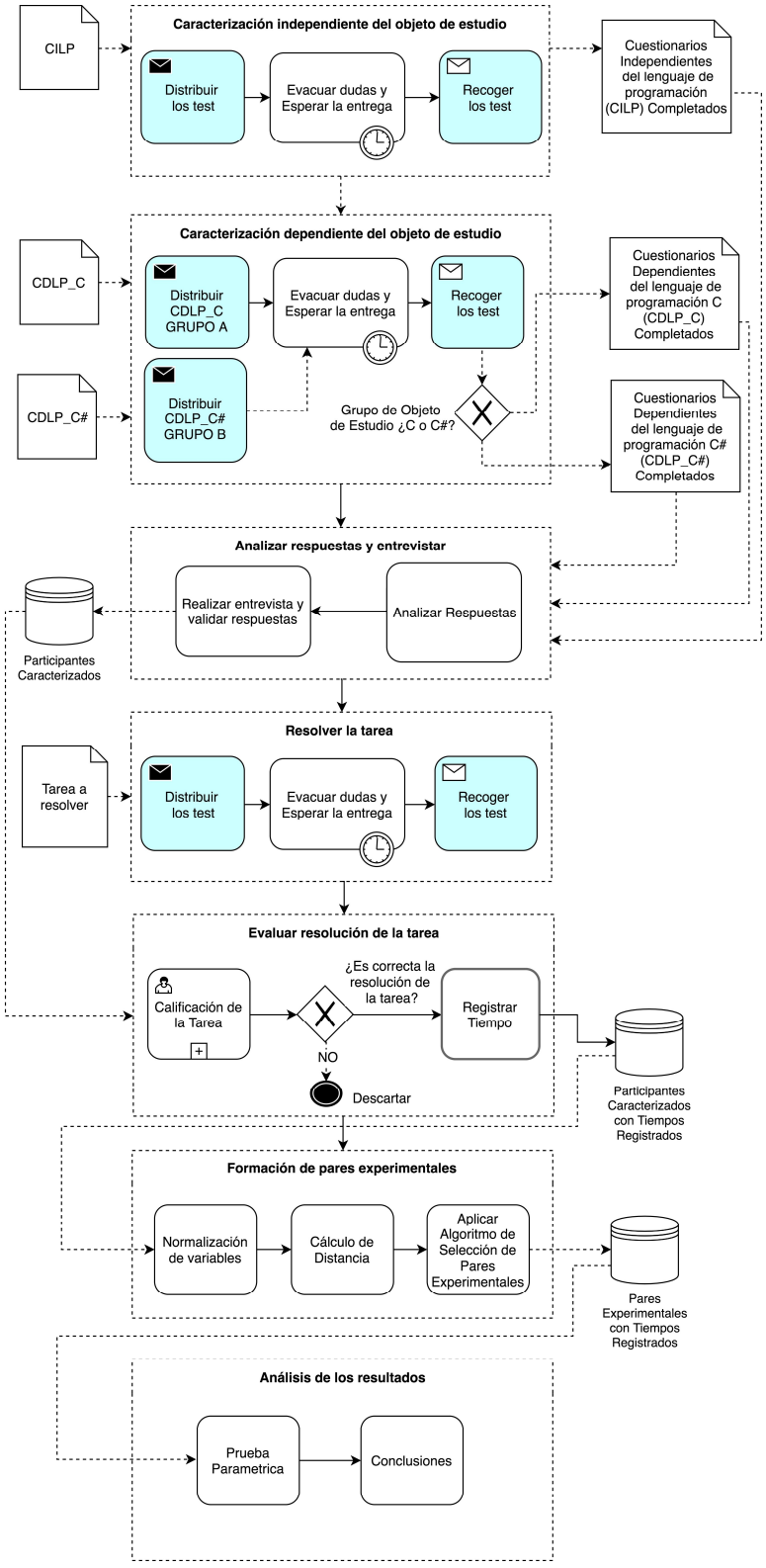

Figura 6.1. Procedimiento de aplicación del protocolo.

ID SUJETO	Conocimiento teórico	Comprensión de código	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica
	CDLP2	CDLP3	CILP3	CILP4	CILP5
1	2.4	0	10.00	0.00	1.64
2	8.1	6.75	9.00	5.00	6.36
3	6.4	4	7.50	5.00	2.73
4	9	9	4.50	5.00	6.36
5	6.3	2.25	6.75	7.00	2.18
6	7.2	6	4.50	9.00	0.67
7	8.1	0	7.50	4.50	1.48
8	5.6	2.25	10.00	4.50	4.43
9	8	4.5	6.75	9.00	2.96
10	6.3	4.5	4.50	4.50	7.64
11	7.2	2.5	6.75	5.00	1.86
12	9	6.75	9.00	9.00	5.45
13	5.6	0	7.50	5.00	7.00
14	6.4	0	9.00	4.50	3.87
23	5.6	0	4	2	6.4
24	7.2	6	6.75	9	2.7
25	8	4.5	10	5	7.6
26	7.2	2.25	10	10	2.5
27	7.2	2.25	2.5	5	5.9
28	6.3	4.5	7.5	5	6.4
29	9	2.5	10	5	4.9
30	3.6	0	5	5	1.8
31	6.3	6.75	7.5	4.5	3.6
32	7.2	4.5	6.75	5	3.4
33	8.1	0	4.5	0	2.5
34	7.2	0	5	0	4.1
35	5.4	2.5	9	4.5	3.8
36	8.1	8	7.5	0	4.5
37	6.3	7.5	7.5	0	6
38	6	2.25	10	0	7.27
39	7.2	6.75	5	4.5	4.3
40	8	2.25	10	4.5	3.2
41	5.4	2.5	7.5	5	4.5
42	8	2.5	7.5	0	4.5
43	8	2.5	4.5	0	7.4

Tabla 6.1. Resultados normalizados de la caracterización de los programadores de C

ID SUJETO	Conocimiento teórico	Comprensión de código	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica
	CDLP2	CDLP3	CILP3	CILP4	CILP5
15	9	9	10.00	4.50	5.91
16	4	5	10.00	0.00	2.86
17	6.3	2.5	7.50	10.00	7.00
18	9	2.25	6.75	5.00	5.18
19	3.6	2.25	5.00	5.00	2.86
20	7.2	6.75	10.00	4.50	5.45
21	8	2.25	7.50	5.00	4.77
22	5.4	6.75	10.00	5.00	7.27
44	8.1	2.5	10.00	4.50	4.43
45	7	2.5	7.50	5.00	4.77
46	6	5	7.5	0	5.73
47	8	5	10	0	5.17
48	5	5	7.50	5.00	2.73
49	8	5	10	5	7.64
50	5	5	10	4.5	1.27
51	4.5	5.25	9	5	4.59
52	6.3	6.75	10	4.5	7.00
53	4.5	5	10	0	2.05
54	6.3	5	5	5	4.86
55	8	4.5	4.50	5.00	7.27
56	5.4	6.75	9.00	5.00	7.64
57	6.3	4.5	6.75	5.00	2.86

Tabla 6.2. Resultados normalizados de la caracterización de los programadores de C#

ID SUJETO	Conocimiento teórico	Comprensión de código	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica	Tiempo
	CDLP2	CDLP3	CILP3	CILP4	CILP5	
2	8.1	6.75	9.00	5.00	6.36	79
4	9	9	4.50	5.00	6.36	85
5	6.3	2.25	6.75	7.00	2.18	121
7	8.1	0	7.50	4.50	1.48	105
9	8	4.5	6.75	9.00	2.96	110
11	7.2	2.5	6.75	5.00	1.86	118
12	9	6.75	9.00	9.00	5.45	115
14	6.4	0	9.00	4.50	3.87	100
24	7.2	6	6.75	9	2.7	81
25	8	4.5	10	5	7.6	118
26	7.2	2.25	10	10	2.5	87
29	9	2.5	10	5	4.9	92
32	7.2	4.5	6.75	5	3.4	122
41	5.4	2.5	7.5	5	4.5	99
43	8	2.5	4.5	0	7.4	121

Tabla 6.3. Resultados normalizados de la caracterización de los programadores de C que completaron la tarea

ID SUJETO	Conocimiento teórico	Comprensión de código	Comprensión de especificación	Comprensión de pseudocódigo	Capacidad Algorítmica	Tiempo
	CDLP2	CDLP3	CILP3	CILP4	CILP5	
15	9	9	10.00	4.50	5.91	38
18	9	2.25	6.75	5.00	5.18	60
20	7.2	6.75	10.00	4.50	5.45	33
22	5.4	6.75	10.00	5.00	7.27	39
45	7	2.5	7.50	5.00	4.77	65
46	6	5	7.5	0	5.73	50
47	8	5	10	0	5.17	58
49	8	5	10	5	7.64	41
50	5	5	10	4.5	1.27	45
51	4.5	5.25	9	5	4.59	30
52	6.3	6.75	10	4.5	7.00	47
54	6.3	5	5	5	4.86	33
55	8	4.5	4.50	5.00	7.27	20
56	5.4	6.75	9.00	5.00	7.64	55
57	6.3	4.5	6.75	5.00	2.86	60

Tabla 6.4. Resultados normalizados de la caracterización de los programadores de C# que completaron la tarea

A continuación se presenta la matriz de distancia entre los programadores de C y los programadores de C# (Tabla 6.5.). En la tabla se encuentran intersecciones pintadas de color negro las cuales representan sujetos que no deben ser tenidos en cuenta a la hora de formar los pares experimentales ya que en por lo menos una de sus dimensiones han presentado una distancia superior al cincuenta por ciento. Los pares experimentales que surgen de aplicar el algoritmo se encuentran resaltados en color verde.

ID	2	4	5	7	9	11	12	14	24	25	26	29	32	33	41	43
15	2.71				7.80		5.15		7.31	4.93			6.37			
18	5.25		4.50	4.52	5.20	3.79	6.43	4.34	6.28	4.74	6.78	3.27	3.39	6.57	3.75	6.00
20	1.70		6.97		6.54	6.47	4.95		6.24	3.25		4.68	4.48		5.36	
22	3.02				7.55				7.16	3.45	8.44	6.05	5.82		5.66	
45	4.90		3.44	4.31	4.99	3.02	6.39	3.15	5.76	4.39	6.04	3.20	2.55	6.83	1.62	6.47
46	5.93	7.71		8.23		6.94		7.15		6.25		6.87	5.72	6.99	5.75	4.70
47	5.52			8.07		7.31		7.11		5.58		5.69	6.29		6.69	
49	2.39				6.98			6.56	7.25	0.50		3.84	5.42		5.39	
50			5.19	6.40	6.55	4.72	7.60	5.89	6.22			5.97	4.52		4.83	
51	4.28	7.60	5.21	7.26	6.05	5.23	6.26		5.70	4.78	6.83	5.38	3.79		3.26	8.54
52	2.21		7.76	9.25	7.42				7.12	2.93		5.48	5.44		5.62	
54	4.96	5.08	4.67	6.79	5.08	4.37	6.54	6.50	5.05	5.97	8.00	6.21	2.50	7.68	3.67	6.39
55	5.11	4.70			6.30			7.41	6.70				4.55	8.25	5.24	5.39
56	2.99	6.32			7.40				7.02	3.58		6.29	5.60		5.49	8.37
57	5.06	6.70	3.09	5.12	4.35	2.41	6.34	5.16	4.37	5.99	6.45	5.10	1.05	7.33	2.84	7.59

Tabla 6.5. Matriz de distancia de los programadores

6.2.2.1. Resolución de la tarea

Como se expresó con anterioridad, la tarea a ser resuelta debe ser la misma para todos los participantes del experimento, independientemente del lenguaje de programación que cada sujeto desee emplear. Dicha tarea puede variar según las características que el investigador desee evaluar, las variaciones pueden corresponderse a múltiples factores como ser:

- **Tamaño de la tarea**: dependiendo de la disponibilidad de recursos con los que cuenta el investigador, las tarea a resolver por parte de los sujetos podría tratarse de un ejercicio el cual busca ser resuelto en unas pocas horas o de una tarea que requiera días o hasta incluso semanas para ser llevada a cabo. En el caso de tratarse de tareas que excedan una jornada de trabajo en un ambiente controlado se deberá arbitrar los medios necesarios para asegurar que cada sujeto lleva adelante la misma por sus propios medios.

- **Tipo de tarea**: al momento de diseñar la tarea el investigador deberá decidir si busca determinar la incidencia o no de un lenguaje de programación en la productividad en tareas de propósito general o si en cambio busca hacerlo para una tarea de propósito específico, por ejemplo, si se busca determinar si la elección del lenguaje A mejora a la productividad de los programadores por sobre el lenguaje B para el desarrollo de un servidor web, la tarea no debería dejar de evaluar el manejo de sockets y threads.

- **Especificación de la tarea**: Se debe procurar especificar la tarea de la manera más clara y descriptiva posible. Según cada grupo de sujetos esto puede variar en la forma de ser llevado a cabo. Es importante dedicar el tiempo necesario para clarificar dudas referidas al enunciado de la tarea, ya que no se desea evaluar la capacidad que tiene un programador en interpretar especificaciones sino el tiempo que a este le insume programar dicha especificación.

En el caso de esta experiencia se optó por una tarea de propósito general la cual no demande más de dos horas en ser resuelta. A continuación se presentan los tiempos expresados en minutos que requirió cada integrante del par experimental en completar la tarea (Tabla 6.6.).

Par Experimental	C#		C	
	ID	Tiempo	ID	Tiempo
A	15	38	12	115
B	18	60	29	92
C	20	33	2	79
D	22	39	26	87
E	45	65	41	99
F	46	50	43	121
G	47	58	14	100
H	49	41	25	118
I	50	43	5	121
J	51	31	24	81
K	52	54	7	105
L	54	51	11	118
M	55	34	4	85
N	56	42	9	110
O	57	51	32	122

Tabla 6.6. Diferencias de tiempo de los pares experimentales de programadores

6.2.2.2. Análisis de los resultados

Los datos tienen que ser validados antes de poder aplicar la prueba paramétrica ya que es posible que en algunos casos los participantes no han podido resolver la tarea de manera correcta o inclusive que en algún caso han abandonado el estudio. Para el caso donde el participante ha resuelto de manera errónea o no ha resuelto en su totalidad la tarea, existen dos alternativas, la primera alternativa es descartar tanto al participante como a su par experimental o bien otra posibilidad es establecer un conjunto de penalidades expresadas en tiempo a cada una de las tareas no resultas o resueltas de manera errónea. La alternativa que incluye penalidades debe de ser cuidadosamente diseñada a efecto de no llegar a un resultado inválido. En este caso todos los

participantes completaron la tarea. Una vez validado los datos, es decir que se conoce para cada par experimental el tiempo insumido por el sujeto que utilizó el lenguaje de programación A y el tiempo del que utilizó el lenguaje de programación B, se puede aplicar la prueba de rangos con signo de Wilcoxon [1945] a efectos de determinar sí existen diferencias significativas entre ambos lenguajes.

Par Experimental	Diferencia d	Rangos de |diferencias|	Rangos con Signo
A	-77	12,5	-12,5
B	-32	0	0
C	-46	3	-3
D	-48	4	-4
E	-34	1	-1
F	-71	10,5	-10,5
G	-42	2	-2
H	-77	12,5	-12,5
I	-78	14	-14
J	-50	5	-5
K	-51	6,5	-6,5
L	-67	8	-8
M	-51	6,5	-6,5
N	-68	9	-9
O	-71	10,5	-10,5

Tabla 6.7. Diferencias de tiempo de los pares experimentales, rangos de la diferencia y rangos con signo

Siendo la suma de los rangos positivos 0 y la suma de los rangos negativos 105, T será la menor de las dos sumas calculadas, encontramos que T = 0. Permitiendo que n sea el número de pares de datos para los que la diferencia d no es 0, tenemos n = 15 y puesto que n <= 30, utilizamos la (Tabla 6.8) para encontrar el valor crítico de 25.

N	0,05	0,02	0,01
4	-	-	-
5	-	-	-
6	0	-	-
7	2	0	-
8	3	1	0
9	5	3	1
10	8	5	3
11	10	7	5
12	13	9	7
13	17	12	9
14	21	15	12
15	25	19	15
16	29	23	19
17	34	27	23
18	40	32	27
19	46	37	32
20	52	43	37
21	58	49	42
22	65	55	48
23	73	62	54
24	81	69	61
25	89	76	68
26	98	84	75
27	107	92	83
28	116	101	91
29	126	110	100
30	137	120	109

Tabla 6.8.. Valor Crítico según el nivel de significancia

6.2.3.3. Conclusiones

Finalmente, luego de evaluar los datos se puede concluir que sí existen diferencias significativas entre los lenguajes de programación evaluados. El valor estadístico de prueba T = 0 es menor al valor crítico de 24, por lo que se rechaza la hipótesis nula, concluyendo que el lenguaje de programación C# es más productivo para resolver la tarea evaluada que el lenguaje de programación C.

7. CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

Esta tesis se enmarca dentro del campo de la ingeniería de software experimental y dentro de este campo es habitual enfrentarse a la necesidad de realizar experimentos con un conjunto reducido de personas las cuales aplican diferentes tratamientos a los objetos de estudio. Realizar comparaciones dentro de pares homogéneos de unidades experimentales no solo aumenta la precisión del análisis sino que también permite controlar gran parte de los efectos no deseados a la hora de realizar un experimento [Juristo & Moreno, 2013]. El objetivo general de este trabajo fue definir un protocolo que permita formar pares experimentales homogéneos de programadores con el propósito de asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. Permitiendo evaluar las características de los lenguajes de programación sin que estas mediciones se vean afectadas por las habilidades o ausencias de las mismas por parte de los programadores. Se presenta un resumen de los resultados del trabajo de investigación teniendo en cuenta los objetivos específicos planteados por este trabajo (sección 7.1) y las futuras líneas de investigación surgidas a partir de él (sección 7.2).

7.1. Conclusiones

Se detallan a continuación los resultados obtenidos en cada uno de los objetivos específicos que ,en su conjunto, han permitido alcanzar el objetivo general.

El primer objetivo específico planteaba realizar un análisis documental en lo que refiere a los beneficios que la formación de pares experimentales aporta en el área de ingeniería de software. Se ha llevado a cabo un análisis documental el cual por un lado permitió comprender que los experimentos realizados en el campo de la ingeniería de software son fuertemente influenciados por las características de los sujetos al igual que ocurre en otras ciencias y por el otro que la realización de comparaciones dentro de pares homogéneos de unidades experimentales a menudo permite aumentar la precisión del análisis.

El segundo objetivo específico planteaba realizar un análisis documental en lo que refiere a los aspectos éticos en relación a las investigaciones en el campo de la ingeniería de software con el fin de tener estos en cuenta en la investigación. Del análisis realizado se destaca la preocupación de muchos autores que destacan que son pocos los trabajos que suelen contemplar los aspectos éticos

de la investigación en el campo de la ingeniería de software experimental. Sieber [2001] sostiene que la investigación empírica en el campo de la ingeniería de software con sujetos humanos no es arriesgada, pero que tampoco está exenta de riesgos y considera que si la ética no se tiene en cuenta en la planificación y realización de la investigación, se puede causar daños a algunos de los implicados en la investigación. Como resultado de este análisis se fijaron las bases para la elaboración del documento de consentimiento (Anexo A), el cual cumple ampliamente tanto con lo planteado por el CONICET [2006] como por Vinson y Singer [2008], se tomaron como modelo las secciones recomendadas por el Comité de Ética de la Investigación perteneciente a la Organización Mundial de la Salud [OMS, 2011].

El tercer objetivo específico planteaba llevar adelante la construcción de los instrumentos que permitan caracterizar a los programadores. Se planteó una caracterización que persigue como objetivo calificar un conjunto de aptitudes del programador a solo efecto de poder encontrar parejas de programadores que se puedan considerar homogéneas. Se ha logrado la construcción de tres instrumentos que permiten caracterizar a los programadores. El propósito que tiene la caracterización es asegurar que dos sujetos de iguales características son indistinguibles en lo que refiere a sus aptitudes como programadores. Por lo cual se optó por analizar un amplio conjunto de aptitudes del programador por entender que realizar una caracterización basada tan solo en unos pocos criterios puede incurrir en graves errores. Para la construcción de las herramientas de caracterización se ha tenido en cuenta el procedimiento general de construcción de un instrumento de medición propuesto por [Sampieri, R. et al., 2010)] y el método para la definición de métricas válidas propuesto por [Genero, M. et al., 2014], adaptando dichos procedimientos a la necesidad de este trabajo.

El cuarto objetivo específico planteaba definir el procedimiento que permite formar los pares experimentales a partir de los datos obtenidos con los instrumentos de caracterización. La elaboración de este procedimiento fue de suma importancia ya que implicaba determinar el mecanismo mediante el cual serían elegidas las parejas de sujetos a ser comparados. Este mecanismo permite tomar las variables que surgen del proceso de caracterización de los programadores, normalizarlas, aplicar penalidades, calcular la distancia que existe entre los programadores y por ultimo seleccionar los pares experimentales basados en dichas distancias.

El quinto objetivo específico planteaba verificar mediante un caso la validez del protocolo de formación de pares experimentales de programadores. Se planteó un experimento en el cual se buscaba determinar si los pares experimentales de programadores formados aplicando el protocolo

presentaban o no diferencias significativas de tiempo a la hora de resolver una misma tarea con un mismo lenguaje de programación. Llegando a la conclusión que en lo que refiere a la formación de los pares experimentales de programadores el protocolo ha funcionado correctamente y por lo tanto se considera que la herramienta tiene un nivel de confiabilidad aceptable, ya que los programadores no han presentado diferencias significativas.

El sexto y último objetivo específico planteaba aplicar el protocolo de formación de pares experimentales de programadores a un caso. El experimento buscó comparar si el lenguaje de programación C presenta diferencias significativas en lo referido a productividad frente al lenguaje de programación C#. Se llegó a la conclusión que sí existen diferencias significativas entre los lenguajes de programación evaluados y que aquellos que utilizaron el lenguaje de programación C# fueron más productivos.

Se puede concluir que en lo que refiere a la formación de los pares experimentales de programadores, el protocolo ha funcionado satisfactoriamente y por lo tanto se considera que tiene un nivel de confiabilidad, validez y objetividad aceptable ya que demostró consistencia entre los resultados

7.2. Futuras Líneas de Investigación

Luego de elaborado el protocolo de formación de pares experimentales y habiendo aplicado este en un caso real con el fin de determinar si un lenguaje de programación en particular afecta o no a la productividad informática , se identifican varias líneas a considerar para trabajos futuros.

Una primera línea consiste en poder desarrollar los instrumentos de caracterización para un conjunto más amplio de lenguajes de programación. Siguiendo lo planteado en este trabajo, se podrán construir y validar nuevos instrumentos de caracterización de programadores que se ajusten a otros lenguajes.

Una segunda línea, dado que se realizó una validación en un sólo caso real, aplicar la herramienta en distintos casos, esto implica poder utilizar el protocolo para formar pares experimentales entre sujetos que utilizan otros lenguajes de programación no incluidos en este trabajo a efectos de poder medir como estos afectan a la productividad informática.

Una tercera línea, aplicar la herramienta en un mismo caso pero en distintos tipos de tareas, esto implica poder utilizar el protocolo para formar pares experimentales entre sujetos que utilizan otros

lenguajes de programación pero midiendo distintos tipos de tareas de manera que se puedan analizar un conjunto más amplio de características de los lenguajes de programación.

Una última línea de investigación, consiste en poder utilizar el protocolo para formar pares experimentales entre sujetos que utilizan el mismo lenguaje de programación a efectos de poder evaluar otras de las herramientas que este utiliza a la hora de programar (editor, control de versiones, depurador , etc.) y determinar cómo estas afectan a la productividad informática.

8. REFERENCIAS

Abdel-Hamid, T. K., & Madnick, S. E. (1989). Lessons learned from modeling the dynamics of software development. Communications of the ACM , 32 (12), 1426–1438.

Abelson, H., Sussman, G. J., & Sussman, J. (1996). Structure and interpretation of computer programs. Justin Kelly.

Ajello, A. M. (2003). La motivación para aprender. En C. Pontecorvo (Coord.),Manual de psicología de la educación(pp. 251-271). España: Popular.

Amazon. (2017). Amazon Cloud Computing. Recuperado el 9 de julio de 2017, de https://aws.amazon.com/es/

ANSI/IEEE. (2007). Draft IEEE Standard for software and system test documentation. ANSI/IEEE Std.

Argimón, J. (2004). Métodos de Investigación Clínica y Epidemiológica. España: Elsevier.

Basili, V. (1993). The Experimental Paradigm in Software Engineering. En H. Rombach, V. Basili, & R. Selby, Experimental Software Engineering Issues: Critical Assessment and Future Directions (pág. 706). Lecture Notes in Computer Science.

Basso, D. (2014). Propuesta de Métricas para Proyectos de Explotación de Información. Tesis de Magister en Ingeniería de Sistemas de Información, Facultad Regional Buenos Aires. Universidad Tecnológica Nacional. Recuperado el 6 de enero de 2017, de http://sistemas.unla.edu.ar/sistemas/gisi/tesis/basso-tesisdemagister.pdf

Bhattacharya, P. &. (2011). Assessing programming language impact on development and maintenance: A study on C and C++. Software Engineering (ICSE) 33rd International Conference (págs. 171-180). IEEE.

Boehm, B. W. (1981). Software engineering economics (Vol. 197). Englewood Cliffs (NJ): Prentice-Hall.

Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., ClarK, B. K., & Horowitz, E. (2000). Software Cost Estimation with COCOMO (Vol. II). Prentice Hall.

Campbell, J. P., McCloy, R. A., & Oppler, S. H. (1993). A theory of performance. En N. Schmitt, & W. C. Bormann, Personnel selection in organizations (págs. 35-70). San Francisco, USA: Jossey-Bass.

Cass, N. (2017). The Top Programming Languages, 2017. Recuperado el 13 de septimbre de 2017, de https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017

CONICET. (2006). Lineamientos para el comportamiento ético en las Ciencias Sociales y Humanidades. Recuperado el 3 de enero de 2017, de www.conicet.gov.ar/wp-content/uploads/RD-20061211-2857.pdf

Coutinho, E., de Carvalho Sousa, F., Rego, P., Gomes, D., & de Souza, J. (2015). Elasticity in cloud computing: a survey. Annals of Telecommunications , 70 (7-8), 289-309.

Creswell, J. (2002). Educational Research: Planning, Conducting, and Evaluating Quantitative. Prentice Hall.

Ericsson. (2017). Erlang programming language. Recuperado el 10 de julio de 2017

Feigenspan, J., Kästner, C., Liebig, J., & Apel, S. (2012). Measuring programming experience. In Program Comprehension (ICPC). IEEE 20th International Conference on (págs. 73-82). IEEE.

Foulds, L. R., Quaddus, M., & West, M. (2007). Structural equation modelling of large-scale information system application development productivity: the Hong Kong experience. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007) (págs. 724-731). IEEE.

Fulgham, B., & Gouy, I. (2009). The computer language benchmarks game. Recuperado el 9 de julio de 2016, de http://benchmarksgame.alioth.debian.org

Genero, M. C.-L. (2014). Métodos de Investigación en Ingeniería del Software. RaMa.

Google Inc. (2017). Google Cloud Platform. Recuperado el 13 de septimbre de 2017, de https://cloud.google.com/

Guardía-Olmos, J., Freixa-Blanxart, M., Turbany Oset, J., & Peró-Cebollero, M. (2007). Análisis de Datos en Psicología. Delta Publicaciones.

Halstead, M. (1977). Elements of Software Science. New York, NY, USA.

Harrison, L., Smaraweera, M., Lewis, D., & Lewis , P. (1996). Comparing Programming Paradigms: An Evaluation of Functional and Object-Oriented Programs. Softw. Eng. Journal , 11 (4), 247-254.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, A., & Cabezas-Isla, F. (2013). Software engineering productivity: concepts, issues and challenges. En Perspectives and Techniques for Improving Information Technology Project Management (págs. 66-79). IGI Global.

IBM. (2017). COBOL Compilers family. Recuperado el 13 de septiembre de 2017, de http://www-03.ibm.com/software/products/en/cobocompfami

IEEE. (1990). Standard Glossary of Software Engineering Terminology IEEE Standard 610.12-1990.

IEEE. (1992). IEEE Standard for Software Productivity Metrics, Std 1045-1992. Institute of Electrical and Electronics Engineers.

IEEE. (1997). IEEE Standard for Developing Software Life Cycle Processes, Std 1074-1997. Institute of Electrical and Electronics Engineers.

Jefferys, J., Hausberger, S., & Lindblad, G. (1954). Productivity in the distributive trade in Europe: wholesale and retail aspects. Organisation for European Economic Co-operation.

Jun, Y. (2007). Towards Adaptive Project Tracking Using Individual Productivity Metrics. 6th International Conference on Machine Learning and Cybernetics, (págs. 19-22). Hong Kong.

Juristo, N., & Moreno, A. (2013). Basics of software engineering experimentation. Springer Science & Business Media.

Kinnersley, B. (2016). The Language List. Recuperado el 24 de abril de 2016, de http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm

Kruger, J., & Dunning, D. (1999). Unskilled and unaware of it: how difficulties in recognizing one's own incompetence lead to inflated self-assessments. Journal of personality and social psychology , 77 (6), 1121-1134.

Lévénez, É. (2015). Computer Languages Timeline. Recuperado el 8 de julio de 2016, de https://www.levenez.com

López, A. V. (2016). Definición de Métricas de Calidad para Productos de Software. XVIII Workshop de Investigadores en Ciencias de la Computación. Entre Ríos: WICC.

Lorenz, M., & Kidd, J. (1994). Object-Oriented Software Metrics.

Loyarte, H., & Novara, P. (2009). Desarrollo e implementación de un intérprete de pseudocódigo para la enseñanza de algorítmica computacional. I Congreso de Tecnología en Educación y Educación en Tecnología.

Maxwell, K. D., & Forselius, P. (2000). Benchmarking software development productivity. Ieee Software , 17 (1), 80-88.

McCabe, T. J. (1976). A complexity measure. IEEE Transactions on software Engineering , 308-320.

Nanz, S., & Furia, C. A. (2015). A comparative study of programming languages in Rosetta Code. 37th International Conference on Software Engineering (págs. 778-788). IEEE Press.

Oktaba, H., Garcia, F., Piattini, M., Ruiz, F., Pino , F., & Alquicira, C. (2007). Software Process Improvement: The Competisoft Project. IEEE Computer , 40 (10), 21-28.

Oliveros, A., & Martinez, S. N. (2012). Aspectos éticos de la investigación en Ingeniería Software que involucra seres humanos. XVIII Congreso Argentino de Ciencias de la Computación.

OMS. (2011). Informed Consent Form Templates. Recuperado el 3 de enero de 2017, de http://www.who.int/rpc/research_ethics/informed_consent/en/

Oracle. (2017). Java. Recuperado el 14 de septiembre de 2017, de https://www.oracle.com/java

Prechelt, L. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl. IEEE Computer , 33 (10), 23-29.

Pressman, R. S. (2015). Software engineering: a practitioners approach. New York, NY: McGraw-Hill Education.

Python Software Foundation. (2017). Python. Recuperado el 15 de septiembre de 2017, de https://www.python.org/

RAE. (2017). Diccionario de la lengua española. Recuperado el 15 de septiembre de 2017, de http://dle.rae.es

Rilling, J., & Klemola, T. (2003). Identifying Comprehension Bottlenecks Using Program Slicing and Cognitive Complexity Metrics. 10th IEEE Working Conference on Reverse Engineering (págs. 115-125). Oregon, USA: IEEE Computer Society.

Riveros, H., & Rosas, L. (1985). El Método Científico Aplicado a las Ciencias Experimentales. México: Trillas.

Rumbaugh, J., Jacobson, I., & Booch, G. (2010). The Unified Modeling Language, Reference Manual. Addison Wesley.

Sammet, J. E. (1972). Programming languages: history and future. Communications of the ACM , 15.

Sampieri, R. H., Collado, C. F., & Lucio, P. B. (2010). Metodología de la investigación. México: McGraw-Hill.

Santrock, J. (2002).Psicología de la educa-ción. México: Mc Graw-Hill.

Scott, M. (2009). Programming language pragmatics. Amsterdam Boston: Elsevier/Morgan Kaufmann Pub.

Sebesta, R. (2012). Concepts of programming languages. Boston: Pearson.

Serrano, M. P. (2002). Un método para la definición de métricas de software. 1er Workshop en Métodos de Investigación y Fundamentos filosóficos en Ingeniería del Software y Sistemas de Información (págs. 65-74). MIFISIS'2002.

Sieber, J. E. (2001). Protecting research subjects, employees and researchers: implications for software engineering. Empirical Software Engineering , 6 (4), 329-341.

Sjoberg, D. H. (2005). A Survey of Controlled Experiments in Software Engineering. IEEE Transactions on Software Engineering , 31 (9).

Soriano, M. A. (2007). Algoritmos Voraces. Recuperado el 3 de Enero de 2017, de Facultat d'Informàtica, U.P.C.: http://www.cs.upc.edu/~mabad/ADA/curso0708/GREEDY.pdf

The Fortran Company. (2017). Fortran. Recuperado el 13 de 9 de 2017, de http://www.fortran.com/

TIOBE Software BV. (2016). TIOBE. Recuperado el 8 de julio de 2016, de http://www.tiobe.com/tiobe_index

Triola, M. (2013). Estadística. (11 ed.). Pearson Educación.

Tsui, F. (2004). Managing software projects. Boston, USA: Jones & Bartlett Learning.

Vernon, M. D. (1973). Motivação humana. Petrópolis, RJ: Vozes

Vinson, N. G., & Singer, J. (2002). Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering , 28 (12), 1171-1180.

Vinson, N. G., & Singer, J. (2008). A practical guide to ethical research involving humans. En Guide to Advanced Empirical Software Engineering (págs. 229-256). Londres: Springer.

Vroom, V. (1964). Work and motivation. New York: Wiley.

Watt, D. A. (2004). Programming language design concepts. John Wiley & Sons.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. Biometrics bulletin , 1 (6), 80-83.

Wilking, D., Schilli, D., & Kowalewski, S. (2008). Measuring the Human Factor with the Rasch Model. Lecture Notes in Computer Science , 5082, 157-168.

Wohlin, C., Runeson, P., Höst, M., & Ohlsson, M. C. (2012). Experimentation in software engineering. Springer Science & Business Media.

Zapponi, C. (2016). Languages at GitHub. Recuperado el 8 de julio de 2016, de http://githut.info/

ANEXO A: FORMULARIO DE CONSENTIMIENTO

CONSENTIMIENTO (FRENTE)

El propósito de este documento es brindarle la información necesaria a efectos de que usted pueda determinar si participar o no de esta investigación. No tiene que decidir en este momento si participará o no en la investigación. Antes de decidir, puede hablar con cualquier persona que usted se sienta cómodo sobre la investigación. Este formulario de consentimiento puede contener palabras que no comprende, de ser así, no dude en solicitar asistencia.

Propósito de la investigación

El objetivo de la investigación es elaborar un procedimiento que permita realizar estudios comparativos sobre la incidencia de los lenguajes de programación en la productividad informática. En la actualidad existe una gran diversidad de lenguajes de programación y a menudo esto dificulta la tarea de seleccionar el lenguaje adecuado para desarrollar una solución determinada.

Selección de participantes

Usted ha sido invitado a participar en esta investigación porque consideramos que su experiencia como programador puede contribuir mucho a nuestra comprensión y conocimiento sobre si el procedimiento planteado para realizar la comparación de los lenguajes funciona debidamente.

Procedimientos

El experimento se divide en dos fases, el entrenamiento y la resolución de una tarea. El entrenamiento tiene por propósito repasar los conocimientos necesarios para realizar la tarea, permitirle interactuar con el entorno de desarrollo donde se llevará adelante la tarea y por último realizar una breve evaluación que permita determinar el nivel en el que usted se encuentra. Previo a resolución de la tarea se comenzará con una detallada explicación de lo que se pretende como solución, se repartirán los enunciados y se le dará la posibilidad de hacer preguntas que tengan por propósito aclara cuestiones referidas a lo solicitado. Agotada la instancia de preguntas, se habilitarán los equipos de los participantes y se dará inicio a la resolución de la tarea. Todos los participantes resolverán la misma tarea (es decir, una implementación de la misma serie de requisitos) independientemente del lenguaje que utilicen.

Duración

La investigación se realiza durante 2 (dos) días en total. Durante ese tiempo, se llevará adelante la jornada de entrenamiento y la de resolución de la tarea, en ambos casos con una extensión máxima de 4 (cuatro) horas.

CONSENTIMIENTO (DORSO)

Confidencialidad

No compartiremos información sobre usted con nadie fuera del equipo de investigación. La información que obtenemos de este proyecto de investigación se mantendrá privada. Nada de lo que nos diga hoy será compartido con nadie fuera del equipo de investigación, y nada será atribuido a usted por su nombr

Resultados

El conocimiento que obtenemos de esta investigación será compartido con usted antes de que esté disponible ampliamente al público. Cada participante recibirá un resumen de los resultados. Luego, publicaremos los resultados para que otras personas interesadas puedan aprender de la investigación.

Participación voluntaria

Su participación en esta investigación es totalmente voluntaria. Es su decisión si participar o no. Usted no tiene que tomar parte en esta investigación si no desea hacerlo. Usted puede dejar de participar en cualquier momento que lo desee.

A quién contactar

Si tiene alg£una pregunta, puede realizarla ahora o más tarde. Si desea hacer preguntas más tarde, puede ponerse en contacto con **Mauricio R. Dávila** [davilamr.80@gmail.com].

Certificado de consentimiento

Me han invitado a participar en esta investigación sobre el "Estudio Comparativo de la Incidencia de los Lenguajes de Programación en la Productividad Informática". He leído la información anterior. He tenido la oportunidad de hacer preguntas al respecto y cualquier pregunta que he hecho ha sido contestada a mi satisfacción. Consiento voluntariamente ser participante en este estudio.

Nombre y Apellido: _____

DNI _____

Fecha _____

Firma _____

ANEXO B: DESCRIPCIÓN DE LA PRUEBA DE RANGOS CON SIGNO DE WILCOXON

En este anexo se describe el funcionamiento de la prueba de rangos con signo de Wilcoxon [1945] la cual es aplicada en el capítulo 6 a efectos de determinar sí existen diferencias significativas en lo que refiere a productividad entre los lenguajes evaluados.

1. Definición

La prueba de rangos con signo de Wilcoxon es una prueba no paramétrica que utiliza rangos ordenados de datos muestrales que consisten en datos apareados. Se usa para probar la hipótesis nula de que la población de diferencias tiene una mediana de cero, de manera que las hipótesis nula y alternativa son las siguientes:

- H0: Los datos apareados tienen diferencias que provienen de una población con una mediana igual a cero.
- H1: Los datos apareados tienen diferencias que provienen de una población con una mediana diferente de cero.

2. Objetivo de la Prueba

La prueba de rangos con signo de Wilcoxon permite analizar la similitud entre un conjunto de datos muestrales apareados donde cada elemento de la población posee un valor experimental que se desea comprobar y un valor de referencia o de control [Guardía-Olmos et al., 2007]. Como resultado de la prueba, es posible aceptar o refutar una hipótesis nula (H0) la cual indica que la población de diferencias entre los valores experimentales y los valores de control tienen una mediana de cero [Triola, M., 2013]. Esto implica que de comprobarse la hipótesis nula mediante esta prueba no paramétrica, se corrobora que no existen diferencias significativas entre los datos comparados.

3. Requisitos de la Prueba:

Para poder llevar a cabo esta prueba se debe cumplir con los siguientes requisitos:

- Todos los valores utilizados deben ser continuos.

- Para cada elemento se encuentra disponible dos valores (datos apareados), los cuales han sido seleccionados en forma aleatoria.
- La población de las diferencias (calculadas a partir de los pares de datos) tiene una distribución que es aproximadamente simétrica, lo que significa que la mitad izquierda de su histograma es, de manera aproximada, una imagen de espejo de la mitad derecha. Por consiguiente, los datos no necesitan tener una distribución normal.

3. Nivel de Significancia

El nivel de significancia es un concepto estadístico que define la probabilidad de rechazar la hipótesis nula cuando esta es cierta [Iglesias Pérez, S/A]. Por consiguiente, este nivel se encuentra asociado a la probabilidad de cometer un error en la prueba y suele ser un valor pequeño (siendo 0,05 el más utilizado).

4. Grado de Confianza de la Prueba

El "grado de confianza" define la probabilidad de no cometer un error al rechazar la hipótesis nula, por lo que dicho grado es calculado como el opuesto del nivel de significancia. Por ejemplo, si se utiliza un nivel de significancia igual a 0,05; entonces el grado de confianza asociado será del 95%.

5. Valores críticos

Siendo n el número de pares de datos para los que la diferencia d no es 0.:

1. Si $n \leq 30$, el valor crítico T se encuentra en la tabla B.1.
2. Si $n > 30$, los valores críticos z se encuentran en la tabla B.2.

6. Procedimiento de la prueba de rangos con signo de Wilcoxon

A partir de los datos recolectados y una vez seleccionado el nivel de significancia (α), se deben llevar a cabo los siguientes pasos esta prueba [Triola, M., 2013]:

- **Paso 1:** Para cada par de datos, calcule la diferencia d restando el segundo valor del primero. Mantenga los signos, pero descarte cualquier par para el que d 0.

- **Paso 2:** Ignore los signos de las diferencias, luego acomode las diferencias de la menor a la mayor y reemplácelas por el valor del rango correspondiente. Cuando las diferencias tengan el mismo valor numérico, asígneles la media de los rangos implicados en el empate.
- **Paso 3:** Agregue a cada rango el signo de la diferencia de la que provino. Esto es, inserte aquellos signos que se ignoraron en el paso 2.
- **Paso 4:** Calcule la suma de los valores absolutos de los rangos negativos. También calcule la suma de los rangos positivos.
- **Paso 5:** Permita que T sea la más pequeña de las dos sumas calculadas en el paso 4. Podría utilizarse cualquier suma, pero para simplificar el procedimiento seleccionamos arbitrariamente la más pequeña de las dos sumas.
- **Paso 6:** Permita que n sea el número de pares de datos para los que la diferencia d no es 0.
- **Paso 7:** Determine el estadístico de prueba y los valores críticos con base en el tamaño muestral.
- **Paso 8:** Cuando plantee la conclusión, rechace la hipótesis nula si los datos muestrales le llevan a un estadístico de prueba que se ubica en la región crítica, esto es, cuando el estadístico de prueba sea menor o igual que el valor (o los valores) crítico(s). De otra forma, no rechace la hipótesis nula.

Valores críticos de *T* para la prueba de rangos con signo de Wilcoxon

	α			
n	.005 (una cola) .01 (dos colas)	.01 (una cola) .02 (dos colas)	.025 (una cola) .05 (dos colas)	.05 (una cola) .10 (dos colas)
5	*	*	*	1
6	*	*	1	2
7	*	0	2	4
8	0	2	4	6
9	2	3	6	8
10	3	5	8	11
11	5	7	11	14
12	7	10	14	17
13	10	13	17	21
14	13	16	21	26
15	16	20	25	30
16	19	24	30	36
17	23	28	35	41
18	28	33	40	47
19	32	38	46	54
20	37	43	52	60
21	43	49	59	68
22	49	56	66	75
23	55	62	73	83
24	61	69	81	92
25	68	77	90	101
26	76	85	98	110
27	84	93	107	120
28	92	102	117	130
29	100	111	127	141
30	109	120	137	152

Tabla B-1. Valores críticos si n ≤ 30 [Triola, M., 2013]

[redacted]	(*continuación*) Área acumulativa desde la IZQUIERDA									
z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495 *	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949 *	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998
3.50 y mayores	.9999									

NOTA: Para valores de *z* por encima de 3.49, utilice 0.9999 para el área.

*Utilice estos valores comunes que resultan por interpolación:

Puntuación *z*	Área
1.645	0.9500
2.575	0.9950

Valores críticos comunes

Nivel de confianza	Valor crítico
0.90	1.645
0.95	1.96
0.99	2.575

Tabla B-2. Valores críticos si n > 30 [Triola, M., 2013]

ANEXO C: DESCRIPCIÓN DE LA TAREA

En este anexo se presenta la tarea que debieron de resolver los participantes del experimento presentado en el Capítulo 6.

1. Tarea

TAREA	
Número de Participante :	
Fecha:	**Lenguaje:**
Hora de Inicio:	**Hora de Fin:**
Se requiere desarrollar una aplicación que permita gestionar los socios de un gimnasio. El programa almacenará los datos de los socios en un archivo del tipo binario, asignándole a cada uno un número de socio incremental e irrepetible según el orden en el que estos son dados de alta. Los datos que se guardarán de cada socio son el nombre, el apellido, el DNI y el número de socio, este último no se le solicita al usuario al momento del alta. El programa contará con una interfaz por consola (Figura 1.), cada opción del menú se encuentra especificada según se indica en el diagrama de casos de uso (Figura 2.).	

Figura 1. Interfaz de la Aplicación

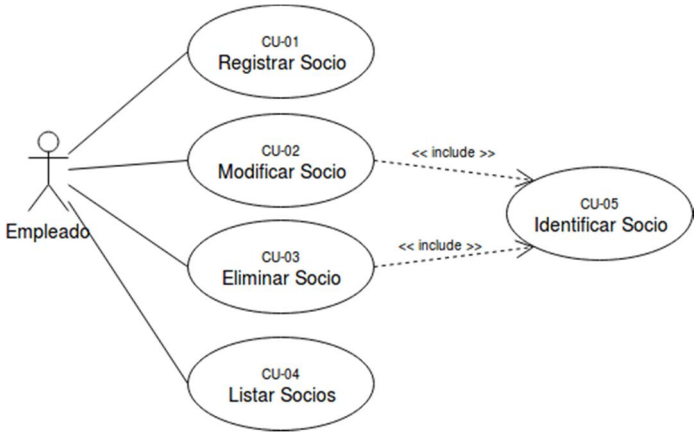

Figura 2. Diagrama de Casos de Uso

CU-01	**Registrar Socio**	
Descripción	El Empleado carga la información de un nuevo Socio en el Sistema.	
Precondiciones	El Empleado cuenta con toda la información y documentación presentada por el nuevo Socio.	
Secuencia Principal	**Paso**	**Acción**
	1	El Empleado ingresa el DNI, nombre y apellidos del nuevo Socio.
	2	El Sistema genera un número de socio para el nuevo Socio.
	3	El Sistema almacena los datos ingresados y el número de socio.
	4	El Sistema muestra un mensaje de éxito, incluyendo el número de socio del nuevo Socio.
Poscondiciones	La información del Socio está cargada en el Sistema.	
Secuencias Alternativas	**Paso**	**Acción**
	*	El Empleado solicita cancelar la operación.
		1 El Sistema cancela la operación.
	1	Alguno de los datos ingresados es inválido.
		1 El Sistema señala el error.
		2 Se continúa desde el paso 1 de la secuencia principal.
	1	El DNI ingresado corresponde a un Socio existente.
		1 El Sistema señala el error.
		2 Se continúa desde el paso 1 de la secuencia principal.
Notas	La información de los Socios se almacenará en el archivo *socios.bin*.	

<table>
<tr><td>CU-02</td><td colspan="3">Modificar Socio</td></tr>
<tr><td>Descripción</td><td colspan="3">El Empleado modifica la información de un Socio.</td></tr>
<tr><td>Precondiciones</td><td colspan="3">El Empleado puede identificar al Socio a modificar.
El Empleado cuenta con la información actualizada del Socio.</td></tr>
<tr><td rowspan="5">Secuencia Principal</td><td>Paso</td><td colspan="2">Acción</td></tr>
<tr><td>1</td><td colspan="2">Se ejecuta el caso de uso Identificar Socio.</td></tr>
<tr><td>2</td><td colspan="2">El Empleado ingresa el nombre y apellidos actualizados del Socio.</td></tr>
<tr><td>3</td><td colspan="2">El Sistema almacena los datos actualizados.</td></tr>
<tr><td>4</td><td colspan="2">El Sistema muestra un mensaje de éxito.</td></tr>
<tr><td>Poscondiciones</td><td colspan="3">La información del Socio está actualizada en el Sistema.</td></tr>
<tr><td rowspan="9">Secuencias Alternativas</td><td>Paso</td><td colspan="2">Acción</td></tr>
<tr><td rowspan="2">*</td><td colspan="2">El Empleado solicita cancelar la operación.</td></tr>
<tr><td>1</td><td>El Sistema cancela la operación.</td></tr>
<tr><td rowspan="2">1</td><td colspan="2">No se identificó ningún Socio existente.</td></tr>
<tr><td>1</td><td>El Sistema señala el error.</td></tr>
<tr><td rowspan="3">2</td><td colspan="2">Alguno de los datos ingresados es inválido.</td></tr>
<tr><td>1</td><td>El Sistema señala el error.</td></tr>
<tr><td>2</td><td>Se continúa desde el paso 3 de la secuencia principal.</td></tr>
<tr><td>Notas</td><td colspan="3">Ninguna</td></tr>
</table>

CU-03	Eliminar Socio	
Descripción	El Empleado elimina la información de un Socio del Sistema.	
Precondiciones	El Empleado puede identificar al Socio a eliminar.	
Secuencia Principal	**Paso**	**Acción**
	1	Se ejecuta el caso de uso *Identificar Socio*.
	2	El Empleado confirma la eliminación del Socio identificado.
	3	El Sistema registra la eliminación del Socio identificado.
	4	El Sistema muestra un mensaje de éxito.
Poscondiciones	La información del Socio está inactiva en el Sistema.	
Secuencias Alternativas	**Paso**	**Acción**
	*	El Empleado solicita cancelar la operación.
		1 El Sistema cancela la operación.
	1	No se identificó ningún Socio existente.
		1 El Sistema señala el error.
	2	El Empleado no confirma la eliminación del Socio.
		1 El Sistema muestra un mensaje.
Notas	Se debe realizar una *baja lógica* (la información del Socio no se puede perder y su número no se puede reasignar).	

CU-04	**Listar Socios**	
Descripción	El Empleado consulta la información de los Socios registrados en el Sistema.	
Precondiciones	Ninguna	
Secuencia Principal	**Paso**	**Acción**
	1	El Sistema muestra un listado de todos los Socios activos, ordenado por apellido y nombre, mostrando para cada uno su número de socio, DNI, nombre y apellidos.
Poscondiciones	Ninguna	
Secuencias Alternativas	**Paso**	**Acción**
	1	No hay Socios registrados en el Sistema.
		1 El Sistema muestra un mensaje.
Notas	Ninguna	

<table>
<tr><th>CU-05</th><th colspan="3">Identificar Socio</th></tr>
<tr><td>Descripción</td><td colspan="3">El Empleado selecciona un Socio registrado en el Sistema.</td></tr>
<tr><td>Precondiciones</td><td colspan="3">El Empleado puede identificar al Socio.</td></tr>
<tr><td rowspan="4">Secuencia Principal</td><td>Paso</td><td colspan="2">Acción</td></tr>
<tr><td>1</td><td colspan="2">El Empleado ingresa el número de socio del Socio a identificar.</td></tr>
<tr><td>2</td><td colspan="2">El Sistema muestra el número de socio, DNI, nombre y apellidos del Socio identificado.</td></tr>
<tr><td>3</td><td colspan="2">El Sistema muestra un mensaje de éxito.</td></tr>
<tr><td>Poscondiciones</td><td colspan="3">La información del Socio está actualizada en el Sistema.</td></tr>
<tr><td rowspan="10">Secuencias Alternativas</td><td>Paso</td><td colspan="2">Acción</td></tr>
<tr><td rowspan="2">*</td><td colspan="2">El Empleado solicita cancelar la operación.</td></tr>
<tr><td>1</td><td>El Sistema cancela la operación.</td></tr>
<tr><td rowspan="3">1</td><td colspan="2">Alguno de los datos ingresados es inválido.</td></tr>
<tr><td>1</td><td>El Sistema señala el error.</td></tr>
<tr><td>2</td><td>Se continúa desde el paso 1 de la secuencia principal.</td></tr>
<tr><td rowspan="3">1</td><td colspan="2">El número ingresado no corresponde a ningún Socio registrado en el Sistema.</td></tr>
<tr><td>1</td><td>El Sistema señala el error.</td></tr>
<tr><td>2</td><td>Se continúa desde el paso 1 de la secuencia principal.</td></tr>
<tr><td>Notas</td><td colspan="3">Ninguna</td></tr>
</table>

ANEXO D: CUESTIONARIO CILP

En este anexo, se presenta el formulario utilizado para la caracterización independiente del lenguaje de programación.

CUESTIONARIO CILP	
Número de Participante :	
Fecha:	**Lenguaje:**
Hora de Inicio:	**Hora de Fin:**
A. De las siguientes alternativas seleccione la que corresponde al nivel educativo máximo del que posee título oficial	
▪ Primaria ▪ Secundaria no técnica ▪ Secundaria técnica ▪ Terciaria ▪ Universitaria de pregrado ▪ Universitaria de grado ▪ Universitaria de posgrado	
B. ¿Cuántas materias o cursos de programación ha realizado hasta el momento? La pregunta se refiere tanto a cursos como a materias en los cuales el objeto de estudio haya sido algoritmia o un lenguaje de programación en particular. Se pueden contabilizar tanto cursos presenciales como los cursos realizados a distancia en tanto y en cuanto hayan sido completados.	
▪ Ninguno ▪ Uno ▪ Dos ▪ Tres ▪ Cuatro ▪ Entre cinco y diez ▪ Más de diez	
C. ¿Cuántos años hace que programa?	

No es relevante si no ha realizado esta actividad de manera profesional.

- Ninguno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

D. ¿En cuántos lenguajes de programación considera tener un nivel regular o superior? Suponiendo la siguiente escala de nivel que un programador tiene en un lenguaje de programación (Muy Malo, Malo, Regular, Bueno y Muy Bueno)

- Menos de uno
- Uno
- Dos
- Tres
- Cuatro
- Entre cinco y diez
- Más de diez

E. Suponiendo que se desea escribir en pseudocódigo un algoritmo capaz de procesar un archivo de texto y en cada línea del mismo se encuentra un numero comprendido en el rango [0 - 99]. El algoritmo deberá recorrer el archivo a efectos de informar por pantalla en qué posición se encuentra el número más alto, en el caso de repetirse dicho número en más de una línea se deberá de informar la primera ocurrencia. La ruta a dicho archivo se le solicitará al usuario al inicio del programa.

1. Indique el resultado que se espera obtener al ejecutar el algoritmo

- La cantidad de ocurrencias del número más grande.
- El valor del número más grande.
- La posición de cada uno de los números con valor igual al número más grande.
- La posición de la primera ocurrencia del número con valor igual al número más grande.
- Ninguna de las anteriores

2. Indique el resultado que se espera obtener al ejecutar el algoritmo

- NO.
- SI, se solicita la cantidad de líneas del archivo.
- SI, se solicita el nombre del archivo.
- SI, se solicita la ruta al archivo.
- Ninguna de las anteriores

3. Indique si existe la necesidad de utilizar una estructura condicional

- NO.
- SI.

4. Indique si existe la necesidad de utilizar una estructura repetitiva

- NO.
- SI.

F. Suponiendo el siguiente bloque de pseudocódigo, ¿Cuál es la salida por pantalla?

```
FOR X = 1 to 100
        IF X%10 == 0 THEN
                PRINT X;
                X=X*2;
        END IF
END FOR
```

- Números que son múltiplo de diez entre uno y cien
- Números que no son múltiplo de diez entre uno y cien
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Números que son múltiplo de diez entre uno y cien al cuadrado
- Ninguna de las anteriores.

G. Suponiendo el siguiente bloque de pseudocódigo, ¿Cuál es la salida por pantalla?

```
NUM X=49;
NUM Z=0;
WHILE X < 50
        PRINT X;
        IF Z== 5 THEN
                X=X+1;
        END IF
        Z=Z+1;
END WHILE
```

- Números entre cero y cuarenta y nueve.
- Solo el número cuarenta y nueve.
- El numero cuarenta y nueve en cinco ocasiones
- El numero cuarenta y nueve en seis ocasiones
- Ninguna de las anteriores.

H. Luego de ser completados los lugares en blanco el programa deberá permitir validar los datos de acceso de un usuario al sistema. El usuario tendrá como nombre de acceso 'admin' y como contraseña '2357'. El programa deberá dar como máximo tres oportunidades. Si el usuario ingresa un nombre de cuenta inválido el mensaje a mostrar será 'USUARIO NO REGISTRADO'. Si ingresa bien el nombre de usuario y la contraseña es incorrecta el mensaje será 'CONTRASEÑA INVALIDA'. Si tanto el nombre como la contraseña son correctos el mensaje será 'BIENVENIDO'. En el caso de exceder la cantidad de intentos el programa deberá finalizar.

```
1  NUM intento = 0;
2  STR usuario;
3   0  password;
4          1
5    2    (intento  3  3)
6  {
7       usuario =     4     ;
8       password = READ();
9
10      IF( usuario 5 "admin")
11      {
12          PRINT "USUARIO NO REGISTRADO";
13          .    6    ++;
14
15      }
16      7
17      {
18          IF( contraseña  8  "2357")
19          {
20              registrado = 1;
21              BREAK;
22          }
23          PRINT "CONTRASEÑA INVALIDA"
24          intento 9 ;
25      }
26 }
27 IF( registrado 10  1)
28 {
29      PRINT "BIENVENIDO"
30 }
```

I. Escriba en pseudocódigo un programa que calcule el sueldo neto de un trabajador quien cobra según las horas trabajadas. El cálculo se realiza de la siguiente forma:

- El valor de la hora es $100
- Las primeras 160 horas a una tarifa fija.
- Las horas extras se pagan a un 50% más de la tarifa fija.
- Los impuestos a deducir de los trabajadores varían según el sueldo mensual.
- Si el sueldo supera los $15000 el trabajador pagara un 10% de impuesto.

El programa deberá solicitar al usuario la cantidad de horas trabajadas e indicar por pantalla el sueldo neto que el trabajador percibirá por sus servicios.

ANEXO E: CUESTIONARIO CDLP_C

En este anexo, se presenta el formulario utilizado para la caracterización dependiente del lenguaje de programación C.

CUESTIONARIO CDLP_C	
Número de Participante :	
Fecha:	**Lenguaje C**
Hora de Inicio:	**Hora de Fin:**
A. ¿Cuántos años de experiencia tiene con el lenguaje de programación elegido? Se debe de contabilizar desde la fecha en que comenzó a utilizar el lenguaje hasta la actualidad.	
▪ Ninguno ▪ Uno ▪ Dos ▪ Tres ▪ Cuatro ▪ Entre cinco y diez ▪ Más de diez	
B. ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional? Se refiere al uso del lenguaje de programación fuera del ámbito académico, entendiendo por esto a la utilización de dicho lenguaje para dar solución a problemas reales.	
▪ Ninguno ▪ Uno ▪ Dos ▪ Tres ▪ Cuatro ▪ Entre cinco y diez ▪ Más de diez	
C. Relacione el nombre de la función con la descripción de la misma.	

scanf ___ strcmp ___ realloc ___ sizeof ___

1. Lee exclusivamente strings de la entrada estándar.
2. Compara dos cadenas de caracteres.
3. Redimensiona memoria previamente reservada con malloc.
4. Retorna la longitud de un array.
5. Copia una cadena de caracteres en otra.
6. Lee datos formateados de la entrada estándar.
7. Reserva memoria de manera dinámica.
8. Retorna el tamaño en bytes ocupado por un tipo de dato.

D. Relacione la palabra reservada con la descripción de su propósito.

define ___ continue ___ typedef ___

1. Se utiliza para crear macros
2. Causa la inmediata salida de una estructura de control.
3. Da un nombre nuevo a cualquier tipo de datos.
4. Declara una variable indicando que su valor es inalterable.
5. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
6. Define un nuevo tipo de dato.

E. ¿Qué es y para qué se utiliza una estructura?

F. ¿Qué es un puntero?

G. ¿En una función a qué se denomina pasaje por valor y pasaje por referencia?

H. ¿Si la variable op tiene valor 'H' cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case 'H' : printf("Hola ");
   case 'J' :  printf("Juan ");
   case 'C' : printf("Chau ");
   break;
}
```

- Hola
- Hola Juan
- Hola Juan Chau
- Ninguna de las anteriores

I. ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
#include <stdio.h>
int main()
{
  int data[5], i;
  for(i = 5;  i > 0; i--)
    *(data + i)  = i;
  for(i = 0;  i < 5; i++)
     printf("%d-", *(data + i));
  return 0;
}
```

- 1-2-3-4-5-
- 5-4-3-2-1-
- 0-1-2-3-4-
- 4-3-2-1-0-
- Ninguna de las anteriores

J. ¿Cuál es el propósito del siguiente bloque de código?

```
int num=50, i=1;
```

```
while( i<=num){
    if( !( num % i)){
        printf("%d", i);
    }
    i++;
}
```

- Muestra números pares desde el 1 al 50
- Muestra los divisores de 50
- Muestra números impares desde el 1 al 50
- Muestra los números que no son divisores de 50
- Error de compilación
- No muestra nada

K. A partir del siguiente código indique qué valor se muestra por la consola

```
int *p, *q;
int x = 4;
p = &x;
q = p;
*q += 3;
printf("%d", *p);
```

- Muestra valores no determinados
- Muestra el número 12
- Muestra el número 4
- Muestra la dirección de x
- Error de compilación
- Muestra el número 7

ANEXO F: CUESTIONARIO CDLP_C#

En este anexo, se presenta el formulario utilizado para la caracterización dependiente del lenguaje de programación C#.

CUESTIONARIO CDLP_C	
Número de Participante :	
Fecha:	**Lenguaje C#**
Hora de Inicio:	**Hora de Fin:**
C. ¿Cuántos años de experiencia tiene con el lenguaje de programación elegido? Se debe de contabilizar desde la fecha en que comenzó a utilizar el lenguaje hasta la actualidad.	
▪ Ninguno ▪ Uno ▪ Dos ▪ Tres ▪ Cuatro ▪ Entre cinco y diez ▪ Más de diez	
D. ¿Durante cuántos años utilizó el lenguaje de programación elegido de manera profesional? Se refiere al uso del lenguaje de programación fuera del ámbito académico, entendiendo por esto a la utilización de dicho lenguaje para dar solución a problemas reales.	
▪ Ninguno ▪ Uno ▪ Dos ▪ Tres ▪ Cuatro ▪ Entre cinco y diez ▪ Más de diez	
C. Relacione el nombre de la función con la descripción de la misma.	

Console ___ Object.Equals ___ Int32.TryParse ___ List.Count ___

1. Compara si dos objetos hacen referencia a una misma instancia
2. Retorna la cantidad de ítems de una lista
3. Intenta convertir un dato a entero. Si lo logra retorna el número convertido.
4. Compara dos objetos por su valor
5. Representa los flujos de salida para las aplicaciones de consola.
6. Intenta convertir un dato a entero. Si lo logra retorna true caso contrario false.
7. Retorna el tamaño en bytes ocupado por la lista
8. Representa los flujos de entrada, salida y error estándar para las aplicaciones de consola.

D. Relacione la palabra reservada con la descripción de su propósito.

using ___ break ___ : (Ej. C1:C2) ___

1. Importa una clase.
2. Produce un salto en la ejecución de código, al pie del bloque que lo contiene.
3. Define una relación de herencia, donde C1 hereda de C2.
4. Importar espacios de nombres (namespaces).
5. Define una relación de herencia, donde C2 hereda de C1.
6. Causa la inmediata salida de una estructura de control.

E. ¿Qué diferencia existe entre un método abstract y uno virtual

F. ¿Qué se puede definir dentro de una interfaz?

G. Si se define class A : B, C. ¿Es esto correcto? De serlo, ¿De qué tipo deberían ser B y C?

L. ¿Si la variable op tiene valor 'H' cuál es la salida por pantalla del siguiente fragmento de código?

```
switch(op)
{
   case "H": Console.WriteLine("Hola ");
   case "J": Console.WriteLine("Juan ");
   case "C": Console.WriteLine("Chau ");
   break;
}
```

- Hola
- Hola Juan
- Hola Juan Chau
- Ninguna de las anteriores

M. ¿Cuál es la salida por pantalla del siguiente fragmento de código?

```
static int main(){
        int max = 5;
        int[] data = new int[max];
        int i;
        for (i = 5; i > 0; i--)
                data[max-i] = i;
        for (i = 0; i < 5; i++)
                Console.WriteLine("{0}-", data[i]);
        return 0;
}
```

- 1-2-3-4-5-
- 5-4-3-2-1-
- 0-1-2-3-4-
- 4-3-2-1-0-
- Ninguna de las anteriores

N. ¿Cuál es el propósito del siguiente bloque de código?

```
int num = 50, j = 1;
while( j <= num){
```

```
            if( (num % j) == 0 ){
                Console.WriteLine("{0}-", j);
            }
            j++;
        }
```

- Muestra números pares desde el 1 al 50
- Muestra los divisores de 50
- Muestra números impares desde el 1 al 50
- Muestra los números que no son divisores de 50
- Error de compilación
- No muestra nada

O. A partir del siguiente código indique qué valor se muestra por la consola

```
Clase1 c1 = new Clase1 (12);
Clase1 c2 = c1;
c1.Dato = 445;
c2.Dato = 4;
Console.WriteLine(c1.Dato);
```

- Muestra valores no determinados
- Muestra el número 445
- Muestra el número 12
- Muestra la dirección de c1 7

ANEXO G: ALGORITMO DE SELECCIÓN DE PAREJAS EXPERIMENTALES

En este anexo se describe el funcionamiento del algoritmo de selección de parejas experimentales el cual es aplicado en el capítulo 5 y 6.

1. Definición

El algoritmo de selección de parejas experimentales tiene por propósito formar pares experimentales integrados por participantes de distintos grupos los cuales se encuentran a la menor distancia euclídea posible entre sí. El algoritmo de selección planteado utiliza una variación del esquema de resolución de problemas denominado método Voraz [Soriano, 2007] y es aplicado a la matriz que describe la distancia existente entre cada uno de los participantes de un grupo y todos los participantes del otro grupo.

El algoritmo sigue una serie de pasos secuenciales y en cada paso toma una decisión, seleccionar el par experimental que presenta la mínima distancia euclídea entre las disponibles, luego en el siguiente paso se encuentra con un problema idéntico, pero estrictamente menor, al que tenía en el paso anterior y allí vuelve a aplicar la misma lógica de selección. A diferencia de un algoritmo voraz clásico, el cual al momento de encontrar más de una solución posible optaría por seleccionar una y descartar las otras, el algoritmo de selección planteado analiza cada uno de los caminos posibles.

2. Procedimiento

A efectos de ejemplificar los pasos llevados a cabo por el algoritmo se toma como punto de partida la siguiente matriz de distancia, la cual que describe la distancia existente entre cada uno de los participantes del grupo A (SA1, SA2, SA3, SA4 y SA5) y los participantes del grupo B (SB1, SB2, SB3, SB4 y SB5). El procedimiento implica encontrar para cada participante del grupo A el par experimental que presenta la mínima distancia euclídea entre las disponibles y en el caso de encontrar más de una solución posible dividir el problema con el propósito de analizar todos los caminos.

GRUPO A

GRUPO B	SA1	SA2	SA3	SA4	SA5
SB1	0	9	1	2	3
SB2	6	8	2	3	9
SB3	1	1	4	9	7
SB4	4	2	5	4	4
SB5	1	10	8	7	5

- **Paso 1:** El algoritmo selecciona el par experimental que presenta la mínima distancia, al no existir más de una opción y debido a que un participante solo puede formar parte de un único par experimental se tachan 'X' de la matriz los valores de los participantes seleccionados.

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	8	2	3	9
SB3	X	1	4	9	7
SB4	X	2	5	4	4
SB5	X	10	8	7	5

- **Paso 2:** partiendo de la matriz resultante del paso anterior, el algoritmo selecciona el par experimental que presenta la mínima distancia, al no existir más de una opción y debido a que un participante solo puede formar parte de un único par experimental se tachan 'X' de la matriz los valores de los participantes seleccionados.

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	3	9
SB3	X	1	X	X	X
SB4	X	X	5	4	4
SB5	X	X	8	7	5

- **Paso 3:** partiendo de la matriz resultante del paso anterior, el algoritmo selecciona el par experimental que presenta la mínima distancia, al no existir más de una opción y debido a que un participante solo puede formar parte de un único par experimental se tachan 'X' de la matriz los valores de los participantes seleccionados.

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	X	X
SB3	X	1	X	X	X
SB4	X	X	X	4	4
SB5	X	X	X	7	5

- **Paso 4:** partiendo de la matriz resultante del paso anterior, el algoritmo selecciona el par experimental que presenta la mínima distancia y al existir más de una opción se analizan ambos caminos.

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	X	X
SB3	X	1	X	X	X
SB4	X	X	X	4	X
SB5	X	X	X	X	5

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	X	X
SB3	X	1	X	X	X
SB4	X	X	X	X	4
SB5	X	X	X	7	X

- **Paso 5:** partiendo de la matriz resultante del paso anterior, el algoritmo selecciona el par experimental que presenta la mínima distancia en ambos caminos.

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	X	X
SB3	X	1	X	X	X
SB4	X	X	X	4	X
SB5	X	X	X	X	5

	SA1	SA2	SA3	SA4	SA5
SB1	0	X	X	X	X
SB2	X	X	2	X	X
SB3	X	1	X	X	X
SB4	X	X	X	X	4
SB5	X	X	X	7	X

- **Paso 6:** Como resultado de aplicar el algoritmo a la matriz de distancia se obtienen tantas tablas como caminos posibles. En dichas tablas figura cada integrante del grupo A apareado con un integrante del grupo B y la distancia euclídea que existe entre ellos.

Sujeto Grupo A	Sujeto Grupo B	Distancia
SA1	SB1	0
SA2	SB3	1
SA3	SB2	2
SA4	SB4	4
SA5	SB5	5
	TOTAL	12

Sujeto Grupo A	Sujeto Grupo B	Distancia
SA1	SB1	0
SA2	SB3	1
SA3	SB2	2
SA4	SB5	7
SA5	SB4	4
	TOTAL	14

- **Paso 7:** La tabla que al sumar todas las distancias arroje el menor valor será es la selección más eficiente.

Sujeto Grupo A	Sujeto Grupo B	Distancia
SA1	SB1	0
SA2	SB3	1
SA3	SB2	2
SA4	SB4	4
SA5	SB5	5