UNIVERSIDAD MADERO PUEBLA
INCORPORADA A LA SEP

Tesis

Desarrollo de Recursos Multimedia e Instrumentos de Apoyo al Proceso de Enseñanza-Aprendizaje para la Materia de Metodología de la Programación

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INGENIERÍA DE SOFTWARE

PRESENTA

ROMÁN ROSAS FLORES

DIRECTOR DE TESIS

MTRA. MARTHA PATRICIA HERNÁNDEZ ZAMORA

"Nuevas respuestas para nuevos tiempos"

2018

Contents

Abstract 7
Capítulo 1: Introducción 8
1.1 Planteamiento del problema 8
1.2 Objetivo General 9
1.3 Objetivos Específicos 9
1.4 Justificación 10
1.5 Alcances y Limitaciones 12
1.6 Organización de la Tesis 14
Capítulo 2. Marco Teórico 15
2.1 Introducción 15
2.2 Algoritmos 16
2.3 Conceptos necesarios para crear un algoritmo 17
2.3.1 Tipos de Datos 17
2.3.2 Variables y reglas para nombrar variables 18
2.3.3 Constantes 18
2.3.4 Operadores 18
2.4 Estructuras de Control 20
2.5 Plan de Configuración Básico 21
2.6 Herramientas de Configuración de Diagramas 21
2.7 Diagrama de Actividades 22
2.8 Prueba de Escritorio 24
2.9 Importancia de C# 25
Capítulo 3. Desarrollo de la Solución Propuesta 26
3.1 Solución Propuesta 26
3.2 Software a utilizar para el desarrollo de la propuesta 26
3.3 Metodología de solución 27
3.4 Video 1. Introductorio 28
3.4.1 Script 29
3.5 Video 2. If 35
3.5.1 If simple 36

3.5.2 Prueba de Escritorio 36
3.5.3 Código en C# 37
3.5.4 Resultado de la ejecución del código en C# 38
3.5.5 If anidado 38
3.5.6 Prueba de escritorio 39
3.5.7 Código en C# 40
3.5.8 Resultados de la Ejecución en C# 41
3.5.9 Script if 42
3.6 Video 3. Switch 58
3.6.1 Prueba de escritorio 59
3.6.2 Código en C# 60
3.6.3 Resultado de la ejecución del código en C# 61
3.6.4 Script 62
3.7 Video 4. For 68
3.7.1 For simple 69
3.7.2 Prueba de Escritorio 71
3.7.3 Código en C# 71
3.7.4 Resultado de la ejecución del código en C# 72
3.7.5 For ascendente en números pares 72
3.7.6 Prueba de escritorio 73
3.7.7 Código en C# 73
3.7.8 Resultado de la ejecución del código en C# 74
3.7.9 For ascendente con números impares 74
3.7.10 Prueba de escritorio 75
3.7.11 Código en C# 75
3.7.12 Resultado de la ejecución del código en C# 76
3.7.13 For Descendente 76
3.7.14 Prueba de escritorio 77
3.7.15 Código en C# 77
3.7.16 Resultado de la ejecución del código en C# 78
3.7.17 For descendente con números pares 78
3.7.18 Prueba de escritorio 79

3.7.19 Código en C# 79
3.7.20 Resultado de la ejecución del Código en C# 80
3.7.21 Script 80
3.8 Video 5. Do-While 91
3.8.1 Do-While simple 93
3.8.2 Código en C# 94
3.8.3 Resultado de la ejecución del código en C# 94
3.8.4 Do-While 95
3.8.5 Código en C# 96
3.8.6 Resultado de la ejecución del código en C# 97
3.8.7 Script 97
3.9 Video 6. While 101
3.9.1 Prueba de escritorio 103
3.9.2 Código en C# 103
3.9.3 Resultado de la ejecución del código en C# 104
3.9.4 Script 105
3.10 Pruebas de efectividad del material multimedia 109
Capítulo 4. Conclusiones, Recomendaciones y Trabajos Futuros 110
4.1 Métricas 110
4.2 Conclusiones 111
4.3 Recomendaciones para aprovechar el tutorial 111
4.4 Trabajos futuros 112
4.5 Lecciones aprendidas 113
Glosario 115
Referencias 118

Índice de Tablas

Tabla 1.	Tipos de datos.	15
Tabla 2.	Operadores aritméticos.	17
Tabla 3.	Operadores relacionales.	17
Tabla 4.	Operadores lógicos.	18
Tabla 5.	Videos del material multimedia.	26
Tabla 6.	Tipos de datos.	31
Tabla 7.	Operadores aritméticos.	32
Tabla 8.	Operadores relacionales.	32
Tabla 9.	Operadores lógicos.	33
Tabla 10.	Prueba de escritorio If simple.	36
Tabla 11.	Prueba de escritorio If anidado.	40
Tabla 12.	Prueba de escritorio Switch.	60
Tabla 13.	Prueba de escritorio For simple.	71
Tabla 14.	Prueba de escritorio For ascendente en números pares.	73
Tabla 15.	Prueba de escritorio For ascendente con números impares.	75
Tabla 16.	Prueba de escritorio For descendente.	77
Tabla 17.	Prueba de escritorio For descendente con números pares.	79
Tabla 18.	Prueba de escritorio While.	103

Índice de Figuras

Figura 1.	Componentes de un diagrama de actividades.	23
Figura 2.	Diagrama de actividades If simple.	36
Figura 3.	Resultado de la ejecución del código If simple.	39
Figura 4.	Diagrama de actividades If anidado.	40
Figura 5.	Resultado de la ejecución del código If anidado.	42
Figura 6.	Diagrama de actividades Switch.	60
Figura 7.	Resultado de la ejecución del código Switch.	63
Figura 8.	Diagrama de actividades For simple.	71
Figura 9.	Resultado de la ejecución del código For simple.	73
Figura 10.	Diagrama de actividades For ascendente en números pares.	73
Figura 11.	Resultado de la ejecución del código For ascendente en números pares.	75
Figura 12.	Diagrama de actividades For ascendente con números impares.	75
Figura 13.	Resultado de la ejecución del código For ascendente con números impares.	77
Figura 14.	Diagrama de actividades For descendente.	77
Figura 15.	Resultado de la ejecución del código For descendente.	79
Figura 16.	Diagrama de actividades For descendente con números pares.	79
Figura 17.	Resultado de la ejecución del código For descendente con números pares.	81
Figura 19.	Diagrama de actividades Do-While simple.	94
Figura 20.	Resultado de la ejecución del código Do-While simple.	95
Figura 21.	Diagrama de actividades Do-While.	96
Figura 22.	Resultado de la ejecución del código Do-While.	98
Figura 23.	Diagrama de actividades While.	103
Figura 24.	Resultado de la ejecución del código While.	105
Figura 25.	Métricas de tiempo y costo	110

Abstract

Por años se ha observado por parte de los docentes de las Ingenierías en Tecnologías de Información e Internet y de Software de la Universidad Madero, que la mayoría de los alumnos de nuevo ingreso no cuentan con conocimientos previos de programación y menos aún de algoritmos, razón por la cual, la deserción en alumnos que sí cuentan con un perfil para la licenciatura, se ha incrementado.

El presente proyecto de investigación tiene como finalidad desarrollar material multimedia que será puesto a disposición de los alumnos mediante una página web que emplea una plantilla para tutoriales creada expresamente para la Universidad Madero. Dicho material permitirá conocer, comprender y aplicar conceptos tales como los de los algoritmos, la traducción de éste al diagrama de actividades, la verificación de que el diagrama está realizado de manera correcta (mediante una prueba de escritorio) y finalmente la traducción del diagrama de actividades a un lenguaje de programación, que en este caso fue el lenguaje C#.

El material elaborado tiene por objetivo que el alumno pueda emplearlo para estudiar a su propio ritmo o como repaso de los temas vistos en clase, por lo que no es necesario contar con conocimientos previos referentes a estos temas. Para ello, se realizó de manera muy minuciosa la investigación y el planteamiento de los conceptos, y se explicaron de manera muy sencilla y concisa para evitar confusiones a quien tenga acceso a esta información.

El objetivo final de este proyecto es que los alumnos puedan contar con recursos adecuados al contenido de la materia de Metodología de la Programación, y que puedan adquirir los conocimientos y habilidades para desarrollar algoritmos correctos, sin causar estrés y logrando avances cognitivos para el uso de los mismos.

Capítulo 1: Introducción

1.1 Planteamiento del problema

La materia de “Metodología de programación”, perteneciente al primer cuatrimestre de las carreras de Ingeniería de Software e Ingeniería en Tecnologías de Información e Internet de la Universidad Madero, tiene como objetivo desarrollar en los alumnos la habilidad de resolver problemas mediante el uso de algoritmos basados en las estructuras básicas de programación, así como la programación del mismo en el lenguaje C#.

Se ha observado que la mayoría de los alumnos de nuevo ingreso en las carreras de Ingeniería de Software e Ingeniería en Tecnologías de Información e Internet no cuentan con conocimientos previos de programación y lógica computacional. Consecuencia de lo anterior, la materia de “Metodología de la programación” no se puede impartir al ritmo planeado, y aunado a esto, no se cuenta con un material de apoyo a la medida y que sea multimedia, el cual auxilie a los alumnos en el repaso puntual de los temas. Un caso concreto, es el de los diagramas de actividades, los cuales se emplean en lugar de los diagramas de flujo con la finalidad de que el alumno aplique las buenas prácticas de un plan de configuración. Otros ejemplos donde el estudiante generalmente requiere de apoyo, es para la correcta ejecución de las pruebas de escritorio con la identificación de los diferentes escenarios y datos a probar, así como la correcta codificación no solo en sintaxis en lenguaje C#, si no también, en la elaboración apropiada del código en cuanto a su indentación y comentarios del mismo.

Por todo lo anterior, es que se observa la necesidad de contar con recursos ad-hoc para la materia de Metodología de la Programación, los cuales apoyen en el proceso de enseñanza-aprendizaje tanto al docente como al alumno. La propuesta realizada por el sustentante, consiste en la elaboración de un material multimedia que contenga temas tales como algoritmos y solución de problemas, el uso de la lógica computacional y procedimental, empleo de pruebas de escritorio y codificación en

C#, el cual permita al alumno repasar a su propio ritmo los temas vistos en clase y reforzar el conocimiento adquirido.

1.2 Objetivo General

Elaborar un tutorial multimedia que apoye a los alumnos de la Ingeniería de Software e Ingeniería en Tecnologías de Información e Internet de la Universidad Madero a comprender la realización de algoritmos, pruebas de escritorio y la codificación de los mismos en C#, mediante la elaboración de una serie de vídeos que permitan al alumno estudiar por cuenta propia y resolver sus dudas acerca de los contenidos vistos en clase.

1.3 Objetivos Específicos

1) Realizar un video con el marco teórico correspondiente:
    a. Algoritmos y solución de problemas usando la lógica computacional
    b. Estructuras de control
    c. Plan de configuración básico
    d. Herramientas de edición de diagramas
    e. Diagrama de actividades
    f. Verificación del cumplimiento de requerimientos
    g. Prueba de escritorio
    h. Codificación en C#
2) Realizar un video independiente por cada estructura de control, siendo éstas:
    a. If e If's anidados
    b. Switch
    c. For
    d. While
    e. Do while

3) Generar un único vídeo para cada estructura de control, que incluya la solución de un problema básico mediante el uso de un algoritmo, una prueba de escritorio con los escenarios correctos y finalmente la traducción del algoritmo en código C#.
4) Realizar las imágenes de los diagramas de actividades en Rational Rose.
5) Subir el material multimedia desarrollado a una página web de la Universidad Madero empleando las plantillas definidas para la elaboración de tutoriales.

1.4 Justificación

Durante años se ha observado que la mayoría de los alumnos de nuevo ingreso de las carreras del área de tecnologías, tales como Ciencias de la computación, Ingeniería en Sistemas, Ingeniería en Informática o afines a éstas, no cuentan con habilidades para resolver problemas utilizando la lógica computacional. Si bien es cierto que los estudiantes en estas licenciaturas o ingenierías tienen cierto enfoque hacia las tecnologías, éstos, en su mayoría aún no cuentan con un nivel mínimo necesario para un correcto desempeño en la carrera que han decidido tomar y a la cual quieren dedicarse.

Un análisis realizado hace 11 años en la Licenciatura en Ciencias de la Computación de la Universidad de Oriente en Cuba permitió observar diferentes deficiencias (Salgado, Alonso, Gorina, Tardo, 2012):

- Limitaciones en la comprensión de situaciones problemáticas que se les plantean y en su respectiva modelación desde la programación.
- Selección y empleo inadecuado de estructuras computacionales que no permiten la verificación y validación de los algoritmos que se conciben y se implementan.
- Imprecisiones en las implementaciones computacionales que se dan a las situaciones problemáticas, las cuales no siempre satisfacen las exigencias originales.

- Escasa destreza en la codificación de procedimientos computacionales en una diversidad de lenguajes computacionales.

Del mismo modo, indican que autores como Guibert, Guittet, Girard (2005), plantean que “los estudiantes que se enfrentan por primera vez a la programación en su proceso de aprendizaje, presentan problemas tales como: no logran desarrollar un modelo viable o estructura que permita resolver el problema, ni describir una estrategia comprensible para la computadora o abstraer los diferentes comportamientos de una tarea en una estrategia que los integre a todos”.

De manera similar también mencionan que los autores cubanos Gonzáles, Estrada y Martínez (2006) proponen que “los primeros conceptos de programación se enseñan a partir de algoritmos y usando pseudocódigos”.

De igual forma hacen mención que Whitfield y otros (2007) de la School of Computer Science of Liverpool Hope University, plantean que “la resolución de un problema computacional es compleja en sí misma, requiriendo habilidades tales como la identificación de los sub-problemas, el reconocimiento de relaciones, situaciones y modelos que permitan desarrollar un algoritmo para la solución y traducción del mismo código ejecutable.”

Por otra parte, mencionan que en un estudio realizado alrededor de varias universidades del mundo por Kaäsboll (2002), asegura en sus informes, que a nivel mundial entre el 25 y 80% de estudiantes que enfrentan en su primer año cursos relacionados a la programación, han reprobado materias o incluso abandonado la carrera por no comprender el contenido que estas implican.

Durante mucho tiempo en las carreras de Ingeniería de Software e Ingeniería en Tecnologías de Información e Internet de la Universidad Madero ha sido notoria la poca demanda que genera en los estudiantes egresados de preparatoria o bachillerato el cursar estas licenciaturas, y de los pocos que llegan a interesarse por las ya mencionadas, en su mayoría no tienen desarrollada esta lógica computacional y procedimental que se requiere para las materias. Estas habilidades poco desarrolladas provoca que algunos de los alumnos de nuevo ingreso se vean

afectados, debido a que los conceptos impartidos en clase no queden completamente comprendidos, provocando así lagunas en el aprendizaje. Esto a la larga genera un problema aún mayor, dado que el conocimiento no adquirido con anterioridad se acumulará y cuando sea requerido en materias de mayor nivel, el alumno no sabrá cómo darle solución; en casos extremos puede traer como consecuencia la baja del alumno de las carreras.

Con lo anterior, se refuerza la necesidad que se tiene por contar con materiales que ayuden a los alumnos que estudian licenciaturas del área de software y tecnologías a desarrollar una lógica computacional y procedimental adecuada, ya que como se presentó previamente, el problema radica en estas habilidades, las cuales no han sido desarrolladas desde grados de estudio anteriores al nivel superior. Adicionalmente, la elaboración de un material multimedia basado en video tutoriales, los cuales expliquen los conceptos básicos de la materia “Metodología de la Programación”, traerán como beneficio que el alumno pueda visualizar el material si los conceptos no quedaron completamente claros durante clase y repetirlos cuando sea necesario.

1.5 Alcances y Limitaciones

- Se utilizará el programa actual de la materia de Metodología de la programación.
- Las animaciones de los videos se realizarán en el software VideoScribe versión 2.3.0.
- Los videos elaborados para este tutorial serán en formato MP4.
- La edición de los videos se realizará en el software Camtasia Studio 8 versión 8.1.
- La grabación de audio se realizará en un estudio especializado para garantizar una mejor calidad del mismo.

- Los materiales multimedia a realizar deberán incluir imágenes, videos y audio.
- Los problemas a resolver con los algoritmos serán de nivel básico.
- La programación del algoritmo se realizará únicamente en el lenguaje de programación C#.
- La herramienta para generar el código C# será el Microsoft Visual Studio Community 2015 versión 14.0.25431.01.
- El tutorial estará disponible para los alumnos a través de una página web la cual se hospedará en un servidor local de la Universidad Madero.
- Se realizarán los vídeos para las siguientes estructuras de control:
    - If e if's anidados
    - Switch
    - For
    - While
    - Do While
- Para cada estructura de control se generará un único vídeo que incluye la solución de un problema básico mediante el uso de un algoritmo, una prueba de escritorio con los escenarios correctos y finalmente la traducción del algoritmo en código C#.
- Las imágenes de los diagramas de actividades estarán elaborados en Rational Rose.
- Se empleará el software IBM Rational Rose Enterprise Edition versión 7.0.0.0 para el modelado de los diagramas de actividades.
- Se elaborará código en Pseint, y solo de forma demostrativa, en otro lenguaje.

1.6 Organización de la Tesis

El presente proyecto de investigación se encuentra organizado de la siguiente forma:

- **Capítulo 1. Introducción**: En este apartado se encuentra el planteamiento del problema, objetivos general y específicos, la justificación para la realización del proyecto, así como sus alcances y limitaciones.

- **Capítulo 2. Marco Teórico**: En esta sección se encuentran los conceptos teóricos básicos sobre algoritmos, sentencias de control, plan de configuración básico, herramientas de edición de diagramas, diagrama de actividades, prueba de escritorio y finalmente se presentará el lenguaje C# como parte de la solución propuesta a un problema.

- **Capítulo 3. Desarrollo de la Solución Propuesta**: En este apartado se muestra la solución propuesta por el sustentante, la cual está conformada por los vídeos y diagramas empleados. Finalmente, se elaborará la plantilla mediante la cual se presentará en la página web el material multimedia desarrollado.

- **Capítulo 4. Conclusiones, Recomendaciones y Trabajos Futuros**: En esta sección se presentan las conclusiones y recomendaciones derivadas de la realización del proyecto, las métricas obtenidas durante la elaboración del mismo, las lecciones aprendidas y posibles trabajos que pueden realizarse en el futuro.

Capítulo 2. Marco Teórico

2.1 Introducción

Para un correcto entendimiento en el desarrollo de un algoritmo, prueba de escritorio y su correcto traslado a código C#, es necesario conocer conceptos básicos sobre estos temas. Es por esta razón que durante el desarrollo de este capítulo se mencionan los conceptos necesarios, referentes a la importancia de la creación de los algoritmos, los cuales involucran “tipos de datos”, “variables”, “constantes” y “tipos de operadores”. También se habla sobre las “estructuras de control”, las cuales serán utilizadas para la creación del material multimedia. Estas estructuras de control serán representadas en “diagramas de actividades”, los cuales son otro aspecto a mencionar para la elaboración de algoritmos, así como las “herramientas de configuración de diagramas”, siendo en este caso “Rational Rose” la que se empleará.

Para poder determinar si el algoritmo que se elaboró es correcto, se emplearán las “pruebas de escritorio”, y por último, se elaborará el código del diagrama en un lenguaje de programación, que en este caso será “C#”.

También es prudente mencionar al “plan de configuración”, el cual es de gran importancia para un correcto estándar de documentación durante el proyecto.

Todos estos temas en conjunto, establecen las bases necesarias para poder comprender cómo se elaboran los algoritmos, conformando así los fundamentos para este tema de investigación.

2.2 Algoritmos

Abellanas & Lodares (1990) mencionan que un algoritmo es un conjunto finito de reglas que dan lugar a una secuencia de operaciones y/o acciones para resolver un problema específico. Es similar a los términos de una receta de cocina, es decir, el algoritmo describe el método mediante el cual se realizará una tarea; consiste en una secuencia de instrucciones, las cuales realizadas de manera adecuada dan lugar al resultado deseado.

Cabe mencionar que los algoritmos no son exclusivamente de la informática y matemática, ya que existen algoritmos que describen toda clase de procesos de la vida diaria.

Abellanas & Lodares (1990) indican que para que un ordenador lleve a cabo una tarea, es preciso indicarle qué instrucciones debe realizar, es decir, se debe describir detalladamente cómo ejecutar la mencionada tarea. Para que esto sea posible, es necesario expresarlo de manera que el procesador entienda lo que significa cada paso y llevar a cabo la operación correspondiente; esto puede lograrse si el algoritmo está expresado a modo que el procesador lo comprenda, es decir, en forma de un programa. El programa se escribe en lenguaje de programación y lo que se escribe en él es el algoritmo previamente diseñado; a esta acción se le conoce como programar. Cada paso del algoritmo se expresa mediante una instrucción o sentencia en el programa, por lo tanto, un programa consiste en una secuencia de instrucciones, de las cuales, cada una especifica ciertas operaciones a realizar por la computadora.

Cabe mencionar que Abellanas & Lodares (1990) destacan que el algoritmo es totalmente independiente tanto del lenguaje de programación como la computadora que se utilice para ejecutar la tarea correspondiente, además de que el diseño de los mismos requiere de creatividad e ingenio y no existen en general reglas para la elaboración de un adecuado algoritmo, en otras palabras, no existe un algoritmo para diseñar algoritmos.

De lo anterior surge la duda del ¿Cómo elaborar algoritmos correctos? ¿En qué basarse para la realización de éstos? Es preciso mencionar que para una correcta elaboración de algoritmos se debe hacer uso del lenguaje, de la lógica y de alguna

técnica para hacer un análisis de la información y posteriormente poder identificar el problema a resolver.

2.3 Conceptos necesarios para crear un algoritmo

Un algoritmo necesita de diferentes componentes para poder realizarlo; a continuación se estarán abordando los que se consideran necesarios.

2.3.1 Tipos de Datos

El tipo de dato es la propiedad que puede tomar determinado valor para poder identificar qué operaciones se le pueden realizar y cómo es representado este valor por la computadora. Algunos de los tipos de datos más utilizados son:

Carácter (char)	Una letra
Texto (string)	Se utilizan para almacenar caracteres y palabras
Entero (int)	Son números enteros, ya sea positivos o negativos
Decimal (double)	Son números enteros más decimales, ya sean positivos o negativos
Lógicos (boolean)	Puede almacenar únicamente 0 y 1, que representan “true (verdadero)” o “false (negativo)”,

Tabla 1. Tipos de datos. (Hernández, 2016)

2.3.2 Variables y reglas para nombrar variables

Para este apartado es importante mencionar en primer lugar qué es una variable, la cual es definida por Hernández (2016) como un “espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato, el cual es modificable a lo largo del algoritmo y programa. Esta variable debe contener una etiqueta o nombre”.

Algunas de las reglas para nombrar variables son las siguientes:

- No puede contener espacios
- No puede comenzar el nombre de la variable con un número
- No puede incluir caracteres especiales
- No pueden llevar tildes
- El nombre debe comenzar con un caracter

2.3.3 Constantes

“Una constante es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato, el cual “no” es modificable a lo largo del algoritmo o programa. Esta constante debe contener una etiqueta o nombre” (Hernández, 2016).

2.3.4 Operadores

“Un operador es un símbolo que tiene una función predefinida, es decir, ya se conoce lo que esta significa y realizará” (Hernández, 2016). Existen tres tipos de operadores, los cuales se mencionan a continuación:

1. Aritméticos: básicamente permiten hacer cualquier operación aritmética que se necesite. En la siguiente tabla se muestran los operadores de los que se dispone:

Operador	Acción
-	Resta
+	Suma
*****	Multiplicación
/	División
%	Módulo

Tabla 2. Operadores aritméticos. (Hernández, 2016)

2. Relacionales: estos operadores permiten evaluar las relaciones entre un par de operandos; a continuación se listan los operadores:

Operador	Acción
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual que
=!	Diferente que

Tabla 3. Operadores relacionales. (Hernández, 2016)

3. Lógicos: estos operadores permiten conectar un par de operandos

Operando	Acción
&&	Y
‖	O
!	Negación

Tabla 4. Operadores lógicos. (Hernández, 2016)

2.4 Estructuras de Control

Las estructuras de control permiten definir el flujo correcto de un programa mediante la correcta ejecución de instrucciones. A diferencia de los algoritmos, ahora se tienen un conjunto de reglas que se deben seguir para la correcta elaboración de los ya mencionados.

Existen tres tipos de estructuras de control (Hernández, 2016):

1. De secuencia: una ejecución sucesiva de una o más operaciones.
2. De selección: dependiendo de una condición se realiza una u otra operación, como es el “if” y el “switch”.
3. De iteración: repetición de una o más operaciones mientras se cumpla una condición, como son el “for, while, do while”.

El presente trabajo se enfocará exclusivamente en las siguientes estructuras de control:

- If: Es la toma de una decisión, para la cual tendremos dos opciones que tomar. El If es un “si” condicional, lo cual significa, “si tal cosa sucede, entonces haz tal cosa” (Arrioja, 2010).
- Switch: esta estructura de control es utilizada para agilizar la toma de decisiones múltiples. Trabaja de la misma manera en que trabajarían los if’s anidados, con la diferencia de que el switch analiza casos y no tanto condiciones como en los if’’s.
- For: a esta estructura de control también se le conoce como estructura cíclica, ya que permite ejecutar una o varias líneas de código de forma repetitiva.
- While: podría decirse que esta estructura de control es un ciclo de preverificación en la cual se evalúa una condición, esta, al ser verdadera, ejecutará el código que tendrá inmerso, de lo contrario, no se ejecutará.
- Do while: esta estructura de control funciona de manera similar que el while, con la diferencia que el código de este ciclo se ejecutará al menos una vez. De ser verdadera la condición se ejecutará más veces.

2.5 Plan de Configuración Básico

El plan de configuración es un documento formal en el cual se especifican todas la configuraciones de un proyecto y se parametrizan todos los documentos en cuanto al tipo de letra, imágenes, interlineados y justificación del documento. En los diagramas se estandariza el color de los diagramas, el color para las notas, nomenclatura y la herramienta para la elaboración de los mismos. De igual forma, detalla el cronograma de actividades, responsabilidades asignadas mediante los documentos correspondientes, los recursos requeridos y las herramientas (UPEDU, 2012).

2.6 Herramientas de Configuración de Diagramas

Cuando se desarrolla software es necesario el uso de artefactos visuales los cuales apoyen el desarrollo óptimo del sistema a desarrollar para el cliente, además de permitir la visualización de los procesos existentes en el sistema así como su detalle para reducir los posibles problemas que este pueda tener. Por este motivo es que existe una metodología tan robusta como lo es UML (Unified Modeling Language/Lenguaje Unificado de Modelado), y por consiguiente también existe una gran variedad de herramientas las cuales hacen uso de esta metodología.

Se puede dividir a las herramientas de edición de diagramas en dos grupos, las versiones gratuitas y las versiones de paga; todas ellas cuentan con la característica que funcionan bajo los estándares de UML, pero con la diferencia que las gratuitas tienen versiones de UML viejas y los símbolos de diagramación no son los actuales.

Algunos ejemplos de software libre son los siguientes:

- ArgoUML: está limitado a trabajar bajo la versión 1.4 de UML. (Tigris.org, s.f.).
- StarUML: cuenta con la ventaja de poder personalizar el software con las necesidades del usuario, pero tiene problemas de integración de diagramas, (Staruml.io, s.f.).

Las herramientas de diagramación libre no proporcionan todo el potencial requerido cuando se realiza un proyecto formal, limitando a la persona encargada de la realización de diagramas a emplear lo que la herramienta le brinda. A diferencia de estas, existen herramientas robustas, las cuales además de estar respaldadas por empresas grandes tales como IBM (International Business Machines) y Microsoft, proporcionan una gran variedad de herramientas dentro del software de diagramación. Son fáciles de utilizar ya que cuentan con una interfaz de usuario intuitiva, permiten una visualización general y específica de los sistemas mediante diferentes tipos de diagramas, la elaboración de los mismos y un conjunto de herramientas las cuales ayudan a un correcto diseño y modelado del sistema. Para IBM se cuentan con los siguientes softwares:

- Rational Rose
- IBM Rational Software Architect Designer

Microsoft cuenta con:

- Microsoft Visio

Para este trabajo se utilizará Rational Rose, ya que la Universidad Madero pertenece a la iniciativa académica de IBM, la cual permite a alumnos y docentes fácil acceso a este software de manera gratuita y con toda la funcionalidad de la versión Enterprise.

2.7 Diagrama de Actividades

El diagrama de actividades es una representación gráfica del algoritmo o proceso que muestra un flujo de trabajo a través de una serie de pasos con los que cuenta el negocio. También son usados para modelar el flujo de control de una actividad a otra, es decir, ilustra el comportamiento y funcionalidad de los sistemas de software. Dentro de los elementos que componen a los diagramas se encuentran (IBM, 2014):

- Acciones: es un paso o actividad que el usuario o software realiza como tarea dada; ésta se representa mediante un rectángulo.

- Nodo de decisión: es una condición que aparece en el flujo dependiendo de la actividad o paso a seguir; ésta se representa por un rombo.
- Flechas de conexión: ayudan a tener un correcto flujo dentro del diagrama.
- Bifurcación: son bastantes útiles cuando se tienen diferentes opciones en determinado paso o actividad; es representada por una línea resaltada (más gruesa que las demás) de manera horizontal.
- Nodo inicial: determina el inicio de la actividad que se estará modelando con el diagrama; es representada mediante un círculo negro.
- Nodo terminal: este nodo determina el final del proceso o actividad modelada; está representada por un círculo negro que además tiene a su alrededor una línea de manera circular de color negro.

A continuación se muestra una imagen referente a lo anterior:

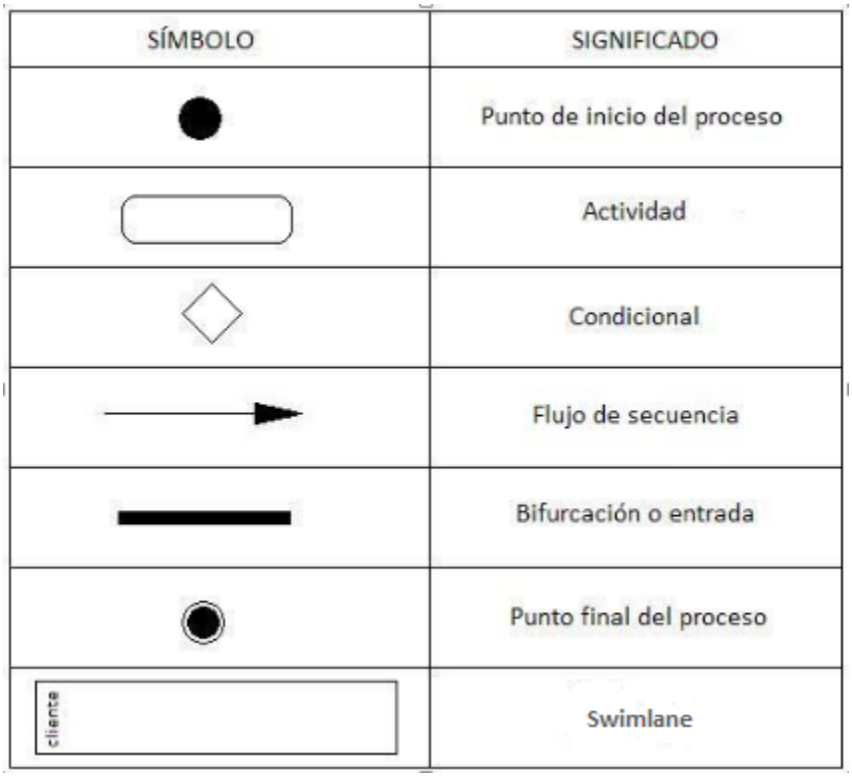

Figura 1. Componentes de un diagrama de actividades. Modificado de: De La Torre (2017)

2.8 Prueba de Escritorio

La prueba de escritorio o también conocida como “casos de prueba” establecen el resultado esperado y el método de evaluación del resultado. Se desarrollan para verificar un comportamiento esperado, el cual, al ser correcto, se le denomina “prueba positiva”. Cuando se dan casos donde el comportamiento no es el esperado, se le nombra como “caso de prueba negativo”, ya que se muestra una condición inaceptable, anormal o inadecuada. Normalmente se desarrollan varios casos de prueba para un algoritmo, con la finalidad de tener un preámbulo general de los diferentes casos que puedan surgir y a los cuales el algoritmo tendrá que hacerle frente (IBM UPEDU, s.f.).

Se han identificado diferentes tipos de casos de prueba, los cuales se listan a continuación:

- Función
- Validación de datos
- Implementación de reglas de negocio
- Flujo de trabajo o control de objetivo de prueba
- Flujo de datos
- Estado del objeto
- Rendimiento (incluida la carga de trabajo, la configuración y el estrés)
- Seguridad y accesibilidad
- Conformidad

Cada caso de prueba describe o representa un conjunto único de entradas o secuencia de eventos que resulta en un comportamiento único por parte del objetivo de la prueba.

2.9 Importancia de C#

C# es un lenguaje de programación de alto nivel el cual cuenta con seguridad de tipos y es orientado a objetos, permitiendo a los desarrolladores elaborar una gran variedad de aplicaciones seguras y sólidas que se ejecutan en el Framework.NET. C# puede usarse para crear aplicaciones cliente de Windows, servicio web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de bases de datos, y muchas otras más.

La sintaxis de C# es muy expresiva, al igual que sencilla y fácil de aprender. Proporciona características importantes para una correcta elaboración de código. C# admite los conceptos de encapsulación, herencia y polimorfismo, todas las variables y métodos, incluido el método “main” el cual es el inicio de todos los proyectos de programación. El proceso de compilación en C# es muy simple a comparación de lenguajes de programación como “C” o “C++” y más flexible que en Java (Microsoft, s.f.).

Capítulo 3. Desarrollo de la Solución Propuesta

3.1 Solución Propuesta

La solución que se propone consiste en la elaboración de un tutorial que se ejecutará mediante una página web que emplea una plantilla desarrollada expresamente para la Universidad Madero.

Dicho tutorial se encontrará alojado en el servidor de proyectos de la Sala de Redes, perteneciente a la Coordinación de Ingenierías en Tecnologías y Software.

El tutorial estará compuesto de materiales multimedia, así como documentos descargables de apoyo para el alumno, tales como el script de los vídeos, los diagramas de actividades de los problemas a resolver y los códigos de programación resultantes de los diagramas.

Con la finalidad de poder lograr el aprendizaje del alumno, se abordarán en primer lugar los conceptos básicos para la elaboración de algoritmos y diagramas de actividades empleando un lenguaje coloquial. La misma estrategia se usará para explicar el funcionamiento del if. Lo anterior permitirá al estudiante que desconoce este tipo de temas, poder incrementar su nivel cognitivo para que primeramente identifique los elementos de un diagrama y de un problema, posteriormente comprenda cómo se usan los componentes del diagrama y su traducción en los diferentes requerimientos del problema a resolver, y por último, que pueda aplicarlos de forma correcta. El resto de los vídeos, continúa empleando un lenguaje coloquial, pero ya no se detalla con tanta especificidad, como en el vídeo del if, pues los elementos básicos ya se trabajaron en dos vídeos previos.

3.2 Software a utilizar para el desarrollo de la propuesta

Para la elaboración de los materiales multimedia se empleó el siguiente software:

- VideoScribe: Se utilizó para realizar las animaciones
- Camtasia: para hacer la edición de los videos.

Cabe aclarar que el audio fue grabado en un estudio y con ayuda de un profesional, con la finalidad de garantizar la calidad del mismo y evitar retrabajos en su creación.

3.3 Metodología de solución

Para el desarrollo de la propuesta se realizará una serie de videos en los cuales se abordarán los temas que son de vital importancia para el correcto diseño y elaboración de algoritmos. Si no se comprenden bien, será complicada e ineficiente la realización del código. Es por esta razón que se diseñó este material, en el cual se dividen diferentes temas en específico, a manera de un aprendizaje progresivo, es decir, se insertará información poco a poco para que el alumno de nuevo ingreso vaya aprendiendo de una forma adecuada.

El material está diseñado a manera de que un estudiante de nuevo ingreso que no cuenta con los conocimientos previos respecto al tema, pueda comprender de lo que se está hablando, ya que el lenguaje empleado durante estos vídeos no involucra términos complicados además de explicar de manera muy detallada los procesos a realizar.

A continuación se muestra la lista de los videos realizados.

Vídeo	Duración (min:seg)	Tamaño (Megas)
1. Introductorio	11:09	35.7
2. If	35:17	77.8
3. Switch	11:19	35.0
4. For	23:58	65.4
5. Do-While	7:56	21.1
6. While	9:38	25.0

Tabla 5. Videos del material multimedia

3.4 Video 1. Introductorio

Para la realización del video introductorio fue necesario recopilar información concisa respecto a lo que involucra la realización de los algoritmos, temas como algoritmos, diagramas de actividades, tipos de datos, variables, constantes, operadores y expresiones, pruebas de escritorio, C# y plan de configuración fueron investigados.

La redacción del capítulo correspondiente al marco teórico de la presente tesis, es muy técnico, por lo que al realizar el video del mismo, se decidió usar un lenguaje más coloquial, debido a que está dirigido a los alumnos de nuevo ingreso quienes no siempre conocen sobre estos temas.

Para esta acción, fue necesario el diseño de un script el cual contuviera la información, esto, con la finalidad de funcionar como la base de la grabación de audio para el video. En este script se cambió del lenguaje técnico a un lenguaje más coloquial para que fuera mayor entendido por los alumnos de nuevo ingreso.

Además, este video (como todos los demás videos) fue realizado de una manera creativa y llamativa, con la finalidad de causar mayor impacto e interés en los alumnos que los visualicen.

Los temas abordados fueron los siguientes:

- Plan de configuración
- Algoritmo y diagrama de actividades
- Tipo de datos, variables y constantes
- Operadores y expresiones
- Rational Rose
- Prueba de escritorio
- C#

3.4.1 Script

Hola! mi nombre es Román, estudié la carrera de Ingeniería de Software en la Universidad Madero, y en esta ocasión te invito a ver una serie de videos que te ayudarán a entender la importancia y creación de algoritmos básicos, verificando que su diseño sea correcto antes de su programación. Posteriormente el algoritmo nos servirá de “receta de cocina” para traducirlo al lenguaje de C#.

En este primer video es indispensable que conozcas los conceptos que comprenden las bases para una buena realización de algoritmos y codificación de los mismos. Los términos sobre los que vamos a hablar son los siguientes:

1. Plan de configuración
2. Algoritmo y diagrama de actividades
3. Tipo de datos, variables y constantes
4. Operadores y expresiones
5. Rational Rose
6. Prueba de escritorio
7. C#

Plan de Configuración

Comencemos con el Plan de Configuración, el cual es un documento formal en el que se especifican y estandarizan formatos de las nomenclaturas, tipos de letras, tamaños de letra, interlineados, justificación del documento, y en el caso específico de diagramas, el color de los diferentes símbolos que componen un diagrama. Esto, con la finalidad de que todos los documentos cumplan con cierto estándar y evitar confusiones con los diferentes entregables a lo largo del desarrollo del proyecto.

Algoritmo

Ahora, pasemos al siguiente punto el cual es el algoritmo. Éste es un conjunto de reglas que dan lugar a determinadas operaciones y/o acciones para resolver un

problema específico. Es similar a los componentes de una receta de cocina. Como sabemos una receta de cocina tiene detallado paso a paso cómo es que se preparará un platillo o postre, así que, el algoritmo describe el método mediante el cual se realizará una tarea, ya que consiste en una secuencia de instrucciones, las cuales realizadas de manera adecuada y ordenada dan lugar al resultado deseado.

Cabe mencionar que el algoritmo es totalmente independiente tanto del lenguaje de programación como de la computadora que se utilice para ejecutar la tarea correspondiente, además de que el diseño del algoritmo requiere de creatividad e ingenio y no existen en general reglas para la elaboración de un adecuado algoritmo, en otras palabras, no existe un algoritmo para diseñar algoritmos.

De lo anterior surge la duda sobre ¿cómo elaborar algoritmos correctos? ¿En qué basarse para la realización de éstos? Es preciso mencionar que para una correcta elaboración de algoritmos se debe hacer uso del lenguaje, de la lógica y de alguna técnica para hacer un análisis de la información y posteriormente poder identificar el problema a resolver y las operaciones a realizar.

Ahora bien, el diagrama de actividades es la representación gráfica de un algoritmo. Pertenece a la metodología UML, estas siglas significan “Lenguaje de Modelado Unificado”, y este diagrama está compuesto por una serie de símbolos específicos, donde cada símbolo tiene un significado y contexto para su uso apropiado, estos son:

- Swimlane: la cual es el área en donde estarán representadas gráficamente las actividades que le corresponde hacer tanto al usuario como al sistema
- Inicio: representa el comienzo del proceso y es definido por un pequeño círculo negro que siempre estará en el swimlane del sistema. Todo diagrama de actividades debe tener un único inicio
- Actividad: es un rectángulo con los costados redondeados. Su objetivo es representar cualquier actividad del proceso, exceptuando una validación. Unos ejemplos del contenido de una actividad pueden ser: declaración e

inicialización de variables, mensajes al usuario, cálculos, lectura de variables, entre otros.

- Condición: esta es representada por la figura de un rombo. Su objetivo siempre será mostrar una condición, la cual evalúa y de acuerdo a la respuesta, tiene dos posibles resultados: cuando la condición es verdadera, afirmativa o un simple si, y la segunda cuando la condición es falsa, negativa o un simple no.
- Flechas: las flechas conectan los diferentes símbolos del diagrama de actividades, visualizando el flujo del diagrama. La flecha debe ser como la mostrada en el ejemplo
- Notas: permite que en el diagrama se comuniquen mensajes adicionales para un mejor entendimiento de las actividades dentro del proceso
- Bifurcación: es una línea gruesa horizontal o vertical de la cual nacen 3 o más flechas de ramificación del flujo, es decir, más posibles caminos a tomar dentro del proceso del diagrama
- Fin: representa el final del proceso y es definido por un círculo que contiene otro círculo negro en su interior. Todo diagrama de actividades, debe tener al menos un final, pero puede contener varios finales, esto dependiendo del diagrama.

Tipos de Datos, Variables y Constantes

Ahora pasaré a explicarte qué son los tipos de datos, variables y constantes. Comenzaremos con los tipos de datos. Al realizar un algoritmo es necesario leer, guardar, imprimir y calcular datos. Los datos son valores que se asignan y que se clasifican por tipos. Los más utilizados son:

Carácter (char)	Una letra
Texto (string)	Se utilizan para almacenar caracteres y palabras
Entero (int)	Son números enteros, ya sea positivos o negativos
Decimal (double)	Son números enteros más decimales, ya sean positivos o negativos
Lógicos (boolean)	Puede almacenar únicamente 0 y 1, que representan "true (verdadero)" o "false (negativo)",

Tabla 6. Tipos de datos. (Hernández, 2016).

Ahora, una variable es un espacio de memoria el cual está reservado para almacenar un valor. Debe tener un nombre y un tipo de dato asociado. El valor que reciba la variable a lo largo del algoritmo y programa puede cambiar. Un ejemplo sería la variable Edad, que guarda la edad de una persona y es un valor de tipo entero. Es importante recalcar que el nombre de la variable debe ser muy específico y dar idea de lo que representa. Otro ejemplo sería una variable que guarda un dato tipo texto, como sería el apellido paterno de una persona.

A continuación te muestro gráficamente cómo se puede entender el concepto de variable:

Nota: Agregar imagen de circulo RAM en el video

Pasemos ahora a hablar sobre las constantes, que al igual que las variables, es un espacio de memoria reservado para almacenar un valor, que corresponde a un tipo de dato, el cual "no" cambia a lo largo del algoritmo o programa. Esta constante debe tener un nombre que lo identifique claramente al igual que la variable.

Operadores y Expresiones

Pasemos ahora a explicar qué son y en qué consisten los operadores, para lo cual se puede decir que un operador es un símbolo que tiene una función predefinida,

es decir ya se conoce lo que realiza y lo que significa. Para explicarte mejor, te mostraré los diferentes tipos de operadores, los cuales se clasifican en tres grupos: los aritméticos, los relacionales y los lógicos (mostrar en pantalla).

A continuación te muestro los aritméticos, los cuales permiten hacer cualquier operación aritmética que se necesite. En la siguiente tabla se muestran los operadores de los que se dispone:

Operador	Acción
-	Resta
+	Suma
*	Multiplicación
/	División
%	Módulo

Tabla 7. Operadores aritméticos. (Hernández, 2016)

Ahora te mostraré los operadores relacionales, los cuales permiten evaluar las relaciones entre un par de operandos, a continuación se listan los operadores:

Operador	Acción
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
==	Igual que
=!	Diferente que

Tabla 8. Operadores relacionales. (Hernández, 2016)

Finalmente se tienen los operadores lógicos, los cuales permiten conectar un par de operandos:

Operando	Acción
&&	Y
||	O
!	Negación

Tabla 9. Operadores lógicos. (Hernández, 2016)

Rational Rose

Es momento de abordar un punto más, el cual consiste en Rational Rose, que es un software de diagramación el cual nos proporciona una gran variedad de herramientas, las cuales son de gran ayuda para la realización de los diferentes tipos de diagramas que se realizan durante el desarrollo de un software. Es fácil de utilizar, ya que cuenta con una interfaz gráfica intuitiva, además, contamos con la ventaja de que Rational Rose funciona bajo los estándares de UML, es decir, al elaborar los diferentes tipos de diagramas, también estamos haciendo uso del lenguaje de creación de algoritmos. Este fue el software en el que se realizaron los diferentes diagramas para estos videos.

Prueba de Escritorio

Hablemos ahora de las pruebas de escritorio. También se les conoce como casos de prueba y éstas tienen como función principal el validar que nuestro algoritmo esté bien realizado. Esto se realiza mediante la validación de un comportamiento esperado, el cual, al ser correcto, lo nombraremos como prueba positiva; de no ser así y que nuestra prueba no tenga el comportamiento esperado, tendremos una prueba negativa. Normalmente se desarrollan varias pruebas de escritorio para un algoritmo, con la finalidad de evaluar los diferentes casos que se puedan presentar, y para verificar que nuestro algoritmo esté bien diseñado y no se tengan inconvenientes en nuestro proyecto.

C#

Finalmente hablemos de C#, el cual es un lenguaje de programación independiente diseñado, desarrollado y estandarizado por Microsoft para generar programas. En este lenguaje de programación se desarrollaron todos los códigos de ejemplo para estos videos.

Bueno, hasta aquí este primer video, el cual consistió en conceptos básicos para un mejor entendimiento de los próximos videos, si alguno de estos conceptos no te quedo claro, te recomiendo ver nuevamente el video o regresarlo en la parte que te genero duda. Una vez que te haya quedado claro, puedes pasar al siguiente video, en el cual explicaremos a detalle el IF. ¡Hasta el siguiente video!.

3.5 Video 2. If

En la elaboración de la primera estructura de control, la cual fue el if, se diseñaron dos ejemplos bastante sencillos, estos, con la finalidad de no complicar en el entendimiento de los algoritmos a los alumnos recién ingresados, ya que, seguramente estos temas serán nuevos para la mayoría de ellos.

Se tomó como primer ejemplo la evaluación de una calificación (ya que todos los alumnos tienen bastante presente la forma de evaluación en las escuelas), con la finalidad de que el alumno entrara en contexto de manera inmediata y comprendiera con mayor facilidad el problema. Para este ejemplo el sistema pedirá al usuario ingresar una calificación, la cual será capturada por el usuario y posteriormente el sistema evaluará la calificación, donde dependiendo de la condición arrojará un resultado, ya sea “El alumno aprobó” o “El alumno reprobó”.

El propósito de este ejemplo es el de mostrar el funcionamiento de la estructura de control If, mediante un ejemplo bastante sencillo y fácil de comprender.

A continuación se mostrará el diagrama del primer ejemplo:

3.5.1 If simple

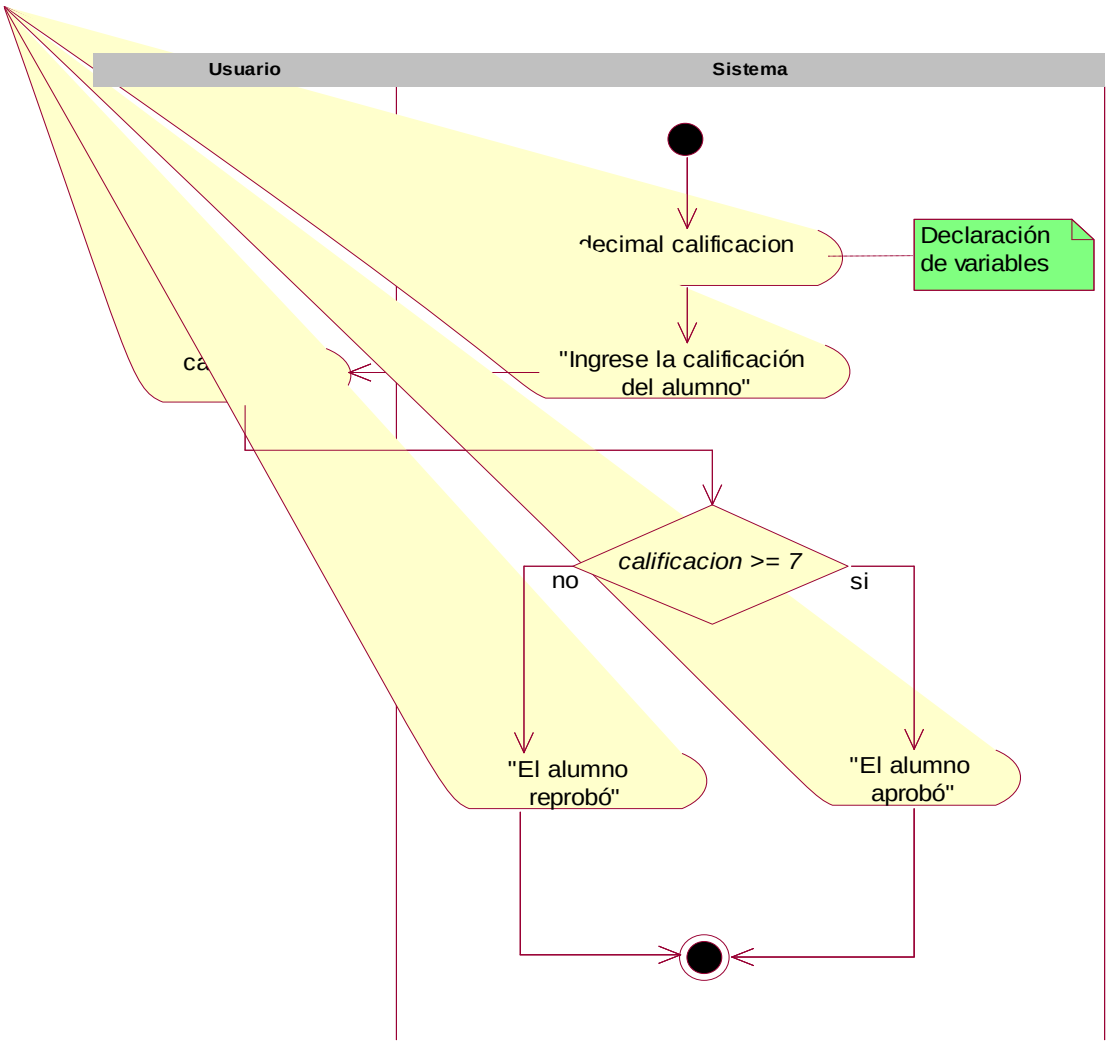

Figura 2. Diagrama de actividades If simple

3.5.2 Prueba de Escritorio

Después del algoritmo visto previamente, lo siguiente es mostrar la forma en que se debe probar el algoritmo; esto se logra con una “prueba de escritorio” o también conocida como “caso de prueba”. En esta prueba de escritorio se emplearon cinco pruebas con calificaciones diferentes, pero, todas ellas con un propósito, el cual, que si bien es demostrar que el algoritmo se diseñó de manera correcta, también se realizó para demostrar que el algoritmo puede tener mejoras. Estas mejoras son referentes a ingresar límites, ya que como se muestra en la siguiente tabla de

pruebas de escritorio, donde se tienen dos pruebas con los valores de 11 y -4, los cuales para el algoritmo que se diseñó funcionan bastante bien, pero no son correctas en el sistema de evaluación en México.

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	6.99	Reprobado
2	7	Aprobado
3	7.1	Aprobado
4	11	Aprobado
5	-4	Reprobado

Tabla 10. Prueba de escritorio if simple

3.5.3 Código en C#

Lo siguiente en este primer video, fue mostrar de qué manera quedaría desarrollado el código en C#, este, basado en el algoritmo que se diseñó al principio del presente apartado. Todo el código fue comentado para que pudiera ser entendido con respecto al algoritmo.

Los comentarios se muestran de color verde:

```
static void Main(string[] args)
        {
            //Declaración de variables
            double calificacion;
            string valor;

            //Solicita la calificación al usuario
            Console.WriteLine("Ingrese la calificación del alumno:");
            valor = Console.ReadLine();
            calificacion = Convert.ToDouble(valor);

            //Verificamos si es una calificación aprobatoria o no
            //Donde: Aprobatoria es de 7 a 10 y reprobatoria de 0 a 6.99
```

```
        if (calificacion >= 7)
        {
            //Mostramos el resultado al usuario
            Console.WriteLine("El alumno aprobó");
        }
        else
        {
            //Mostramos el resultado al usuario
            Console.WriteLine("El alumno reprobó");
        }
    }
```

3.5.4 Resultado de la ejecución del código en C#

Finalmente, se mostró la ejecución del código, para esto, se emplearon dos ejemplos, uno para la calificación de 6.99 y la otra de 7.

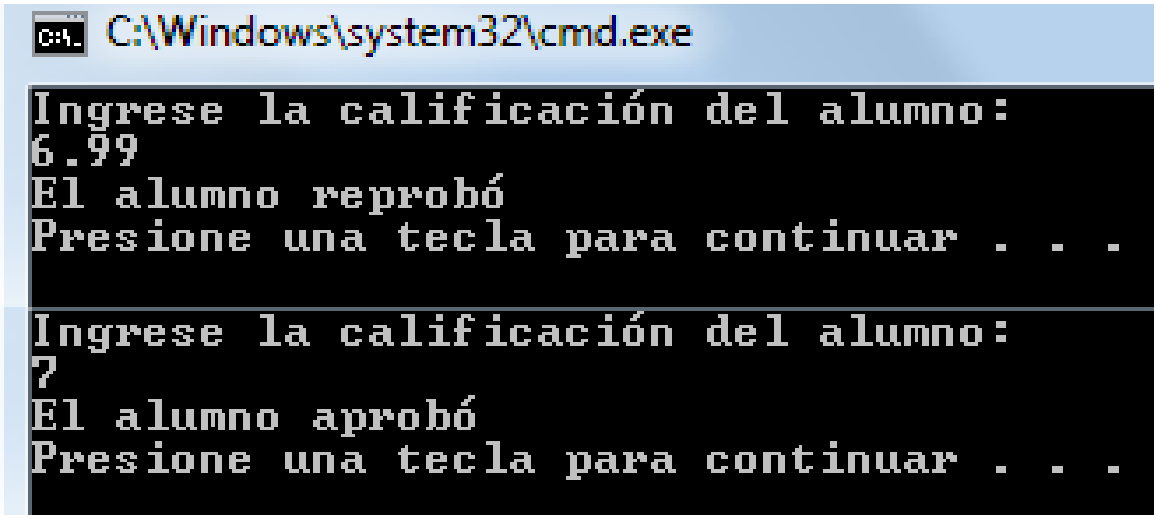

Figura 3. Resultado de la ejecución del código If simple

3.5.5 If anidado

En este segundo ejemplo, se continúa con el formato del ejercicio anterior, pero en esta ocasión se modifica el algoritmo para validar precisamente los datos frontera,

con la finalidad de que cuando el usuario ingrese datos como 11 y -4 envíe un mensaje diciendo que es un valor no válido.

El propósito de este segundo ejemplo es demostrar que la estructura de control if puede estar anidada, es decir, pueden emplearse de forma consecutiva entre sí y optimizar de esta manera al algoritmo. Como se muestra, ahora se tienen dos condiciones dentro de un rombo el cual (para este caso) representa a un if.

La explicación detallada de este ejemplo y del anterior se redacta en el script de la estructura de control if.

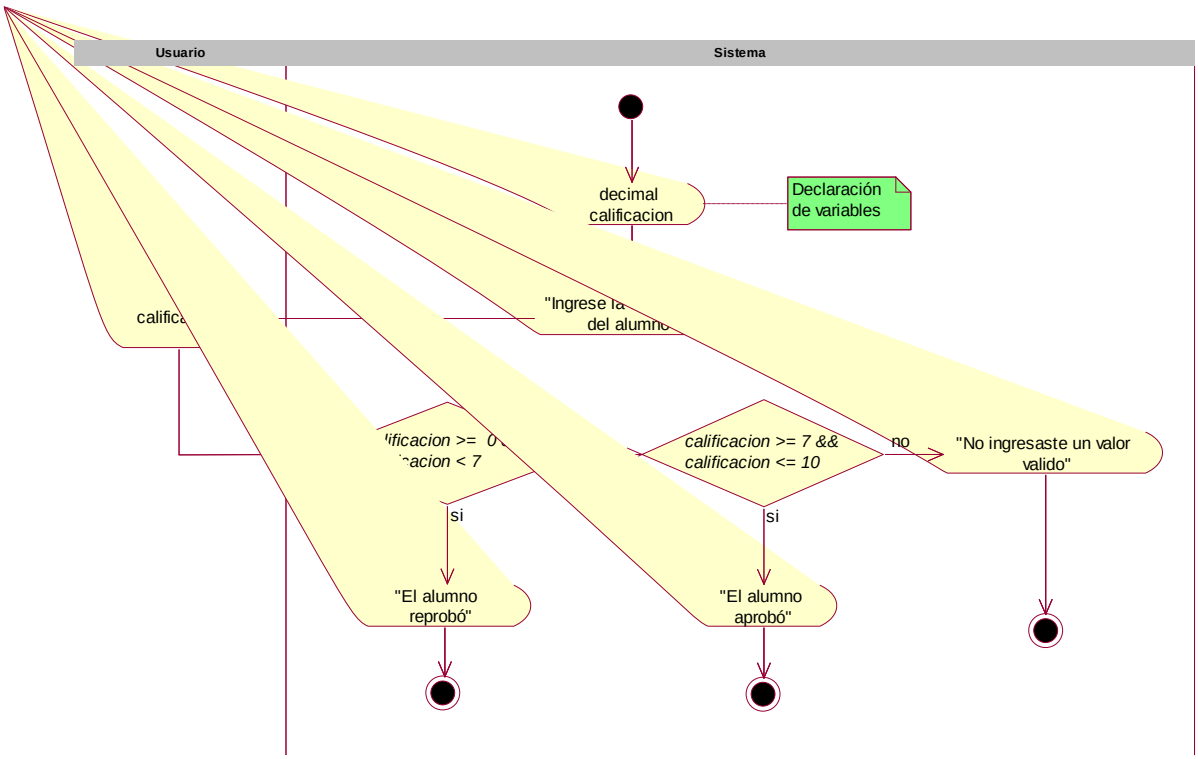

Figura 4. Diagrama de actividades if anidado

3.5.6 Prueba de escritorio

Para este segundo ejemplo se realizaron seis pruebas de escritorio, con la finalidad de poder corroborar valores frontera como lo son el 0 y el 10, valores aprobatorios

como lo fue el 7, que además es el valor utilizado en la condición del if, además de valores no permitidos, como lo fueron el -1 y el 11.

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	0	Reprobado
2	6.99	Reprobado
3	7	Aprobado
4	10	Aprobado
5	-1	Error
6	11	Error

Tabla 11. Prueba de escritorio if anidado

3.5.7 Código en C#

A continuación se muestra el código para este segundo algoritmo, el cual se diseñó con base en el mismo. Todo el código ha sido comentado para mejor entendimiento del alumno que visualice el video. Estos comentarios son de color verde.

```
static void Main(string[] args)
        {
            //Declaración de variables
            double calificacion;
            string valor;

            //Solicita la calificación al usuario
            Console.WriteLine("Ingrese la calificación del alumno:");
            valor = Console.ReadLine();
            calificacion = Convert.ToDouble(valor);

            //Verificamos si es una calificación aprobatoria o no
            //Donde: Aprobatoria es de 7 a 10 y reprobatoria de 0 a 6.99
```

```
        if (calificacion >= 0 && calificacion < 7)
        {
            //Mostramos el resultado al usuario
            Console.WriteLine("El alumno reprobó");
        }
        else if (calificacion >= 7 && calificacion <= 10)
        {
            //Mostramos el resultado al usuario
            Console.WriteLine("El alumno aprobó");
        }
        else
        {
            //Mostramos el resultado al usuario
            Console.WriteLine("No ingresaste un valor valido");
        }
    }
```

3.5.8 Resultados de la Ejecución en C#

Finalmente, se muestra la ejecución del código; en esta ocasión para tres pruebas de escritorio, las cuales fueron los valores de 6.99, 7 y -1.

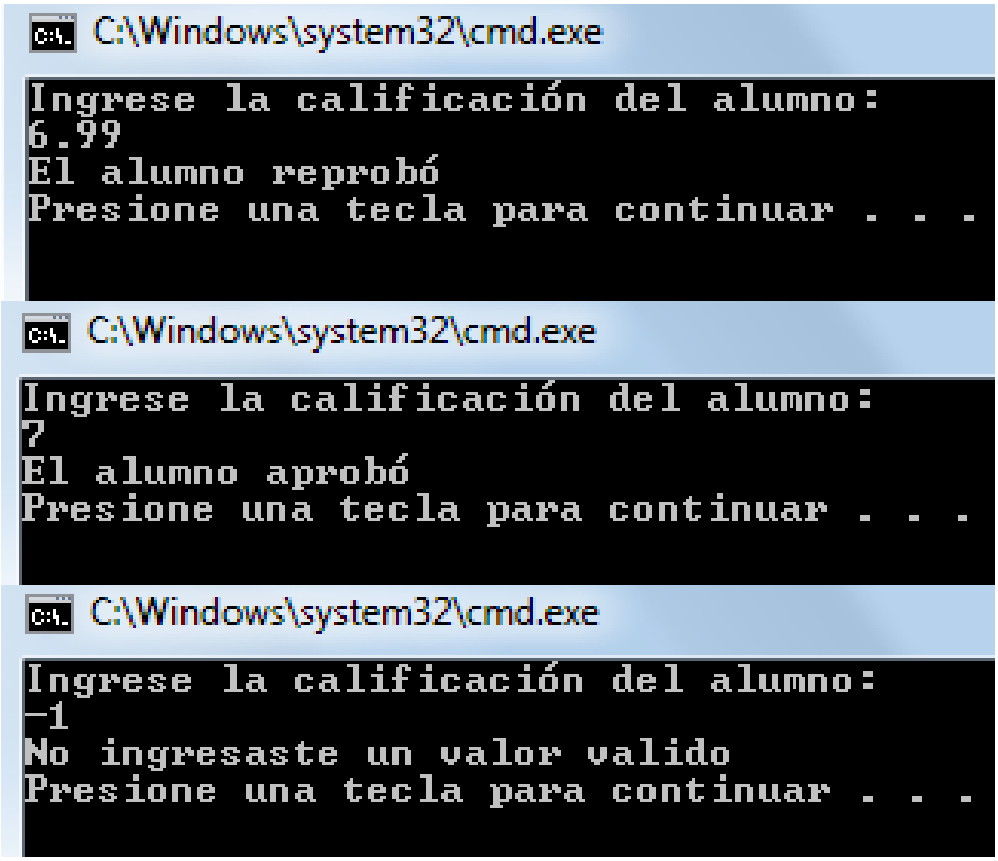

Figura 5. Resultado de la ejecución del código if anidado

3.5.9 Script if

Como tal, en el script del if se ha conjuntado todo el trabajo previo del primer video con la información introductoria, ya que en el desarrollo del script, se retoman los conceptos necesarios de manera precisa y en el momento adecuado, esto con el propósito de recordarle al alumno los conceptos. De este modo, el estudiante puede relacionar los conceptos con los elementos que contiene el diagrama de actividades y qué significan cada uno de ellos.

A continuación se muestra el script de este primer video if:

Hola! mi nombre es Román, estudié la carrera de Ing. de Software en la Universidad Madero y en este segundo video te hablaré sobre las estructuras de control, las cuales permiten definir el flujo correcto de un programa. Esto es posible mediante

la correcta ejecución de instrucciones, a diferencia de los algoritmos, ahora se tienen un conjunto de reglas que se deben seguir para la correcta elaboración de dichas estructuras.

Existen tres tipos de estructuras de control:

4. De secuencia: en este de tipo de estructuras existe una ejecución sucesiva de una o más operaciones.
5. De selección: en este caso, dependiendo de una condición, se realiza una u otra operación, como es el caso del “if” y el “switch”.
6. De iteración: se emplea cuando se requiere la repetición de una o más operaciones mientras se cumpla una condición, como son el “for, while, do while”.

En este segundo video comenzaremos con la primera estructura de control de selección que es el IF. De manera muy sencilla podemos decir que el IF representa la toma de una decisión, en la cual tendrás que elegir entre dos opciones. Por mencionar un ejemplo bastante sencillo, imagina que vas manejando un auto y que observas la luz del semáforo para determinar qué debes hacer, es decir, debes tomar una decisión acerca de detenerte o continuar avanzando. Para mostrar cómo se podría asemejar a un IF donde se tomaría una decisión, tendríamos que plantearlo de la siguiente forma:

Si luz = verde entonces

    Avanzar (que corresponde a una respuesta afirmativa o verdadera)

Si no

    Detenerse (que corresponde a una respuesta negativa o falsa)

Como se observa, en el primer renglón se muestra la condición o decisión que debe tomarse. El renglón siguiente a la condición, siempre corresponderá a una respuesta afirmativa, verdadera o simplemente un “sí”. El renglón que se encuentra en seguida del “si no” corresponde a una respuesta negativa, falsa o un simple “no”. A continuación, mostraré dos ejemplos básicos en los cuales se usa el IF.

Para el primero tenemos el siguiente planteamiento:

Problema 1: Realizar un algoritmo gráfico en forma de diagrama de actividades que solicite al usuario una calificación y en base a su respuesta, muestre si el alumno está aprobado o reprobado. La calificación podrá tener hasta dos decimales y para poder decir si un alumno aprobó, la calificación mínima deberá ser 7.

Antes de comenzar recordemos lo que es un diagrama de actividades: es una representación gráfica del proceso, en donde se detalla paso a paso lo que se realiza.

Para construirlo llevaremos a cabo los siguientes pasos:

- Paso 1: Se colocan dos swimlanes (un swimlane es el área en la cual estarán representadas gráficamente las actividades que le corresponde hacer tanto al usuario como al sistema)
- Paso 2: Nos colocamos en el lado del swimlane del sistema para crear el inicio del diagrama, el cual está representado por un círculo negro y siempre estará en el swimlane del sistema
- Paso 3: Se requiere declarar las variables (recordemos que una variable es un espacio de memoria virtual, reservada para almacenar un valor y es modificable a lo largo del algoritmo). Para este caso tenemos una variable a la cual llamaremos calificación; esta variable (así como todas las variables) se escribe sin acento, ya que las variables no llevan acento y en este caso concreto es de tipo decimal. Esta declaración de variable se escribe en un rectángulo con los costados redondeados, el cual significa que en él debe estar escrita una actividad, y una declaración de variable es una actividad. Lo siguiente es seguir el flujo correcto de nuestro proceso a realizar, éste se describe con flechas las cuales muestran la dirección que debe seguirse.

  Hagamos referencia ahora al plan de configuración, el cual habíamos dicho que es un documento formal en el que se especifican todas las configuraciones del proyecto, es decir, se estandarizan los tamaños y tipo de letra del documento, los títulos y la forma en que deben estar

nombrados los diagramas e imágenes que se empleen. Cuando se hace referencia al plan de configuración desde el diagrama de actividades, se pueden poner notas donde se coloquen comentarios, como en este caso “Declaración de variables”, y éste será de color verde limón (como todas las notas) y se liga a la actividad con una línea punteada dentro del diagrama que estamos construyendo.

- Paso 4: el sistema solicita al usuario ingresar una calificación. Como se trata de una actividad, también se coloca en un rectángulo con los costados redondeados
- Paso 5: ahora, es necesario indicar que lo siguiente a realizar le corresponde al usuario, es por esta razón que el flujo de la flecha se orienta hacia la izquierda, hacia el swimlane del usuario, donde deberá ingresar una calificación. Para ello se coloca en el diagrama la variable calificación como actividad, la cual almacenará el valor ingresado. Esta actividad debe colocarse en el swimlane del usuario, ya que es él quien proporciona la información; esto es lo único que hará el usuario. Es por esta razón que en el diagrama se visualiza que la flecha regresa hacia el lado derecho, donde se encuentra el swimlane del sistema, ya que lo que nos falta realizar del proceso continuará allí.
- Paso 6: una vez que tenemos la calificación es necesario identificar si el valor cae en el rango de aprobada o reprobada. Esto se puede hacer mediante una condición que se coloca dentro de un rombo que representa a un if.

  La condición que usaremos será:

  Si calificación >= 7 se enviará el mensaje

  “El alumno aprobó”

  Si no

  “El alumno reprobó”

  Traduciendo esto al diagrama, tenemos que dentro del rombo escribiremos: “calificación >= 7” y las dos posibles respuestas se

representarán como salidas del rombo en caso de que la respuesta sea “SI es mayor o igual a 7” o “NO es mayor o igual a 7”

- Paso 7: Ahora hay que colocar en el diagrama lo que sucederá cuando la respuesta es “SI” y cuando la respuesta es “NO”. Si la respuesta es “SI, Afirmativa o Verdadera” deberá enviar un mensaje que indique “El alumno aprobó”. Si la respuesta es “No, Negativa o Falsa” deberá enviar el mensaje “El alumno reprobó”, tal como se mostró que se debía hacer en el paso anterior. Es importante aclarar que las salidas del rombo pueden colocarse en cualquiera de sus vértices libres de otros flujos, lo cual también significa que la continuación del diagrama puede ser hacia cualquier orientación, es decir, “Arriba, Abajo, Izquierda o Derecha” dependiendo de los vértices que tenga libres y siempre dentro del swimlane correspondiente. Es muy importante aclarar que siempre deben colocarse las palabras “si” y “no” a los flujos que salgan del rombo que representa al IF, de lo contrario, no se podrá saber qué procedimiento corresponde a cada una de las posibles salidas.
- Paso 8: Como observamos, dependiendo del resultado de la condición, el algoritmo solo tomará un camino, el de “SI” o el de “NO”, nunca podrá tomar los dos al mismo tiempo. Esto significa que dado el requerimiento de lo que debía hacer el algoritmo, que era definir si una calificación era aprobada o reprobada, una vez que ha determinado a cuál de las dos pertenece, el diagrama deberá finalizarse, es decir ya no habrá más actividades por hacer. Por lo tanto, se deberá finalizar el diagrama para cada uno de los flujos. Un diagrama debe tener al menos un final, y en este caso ambos flujos pueden apuntar al mismo final, o finalizarse cada uno por separado.

  Para finalizar un diagrama se coloca un círculo que contiene otro círculo negro en su interior.

A continuación verificaremos que nuestro algoritmo esté correcto. Para ello utilizaremos la prueba de escritorio (la cual es una tabla que muestra los diferentes valores que va a tomar nuestra variable y el resultado que se obtendrá cada vez que se pruebe el algoritmo). En este caso realizaremos 5 pruebas de escritorio con la finalidad de verificar el algoritmo con diferentes datos, los cuales escogeremos al azar:

- Para la primera prueba usaremos una calificación de 6.99, que es una **calificación** con dos decimales y menor a la mínima aprobatoria
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 6.99 (que es el valor con el que decidimos probar el algoritmo)
    - El valor de 6.99 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 6.99, es decir: Si 6.99 es mayor o igual a 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como 6.99 no es mayor o igual a 7, el algoritmo seguirá por el flujo de “No”, y enviará el mensaje “El alumno reprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio. Debido a que pudimos probar que una calificación de 6.99 es menor a 7 (que es la calificación mínima aprobatoria) y que el algoritmo reportó que está reprobado, podemos decir que la prueba fue correcta así como nuestro algoritmo. Esta es la finalidad de la prueba de escritorio: que nos pueda ayudar a determinar si nuestro algoritmo es correcto.

- Para la segunda prueba la calificación que usaremos será 7, que es la mínima aprobatoria que se definió en el planteamiento del problema y que usamos en la condición del IF
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 7

- El 7 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 7, es decir: Si 7 es mayor o igual a 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como 7 es mayor o igual a 7, el algoritmo seguirá por el flujo de “Si”, y enviará el mensaje “El alumno aprobó”. Posteriormente, finaliza el algoritmo correctamente y por lo tanto, nuestra prueba de escritorio.

- Ahora para la tercera prueba, la calificación será de 7.1
  - Primero el sistema solicita una calificación al usuario
  - Después el usuario ingresa el 7.1
  - El 7.1 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 7.1, es decir: Si 7.1 es mayor o igual a 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como 7.1 es mayor o igual a 7, el algoritmo seguirá por el flujo de “Si”, y enviará el mensaje “El alumno aprobó”. Posteriormente, finaliza el algoritmo correctamente y por lo tanto, nuestra prueba de escritorio.

- Para la cuarta prueba la calificación será 11
  - Primero el sistema solicita una calificación al usuario
  - Después el usuario ingresa el 11
  - El 11 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 11, es decir: Si 11 es mayor o igual a 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como 11 es mayor o igual a 7, el algoritmo seguirá por el flujo de “Si”, y enviará el mensaje “El alumno aprobó”. Posteriormente, finaliza el algoritmo satisfactoriamente y por lo tanto, nuestra prueba de escritorio.
- Para la quinta prueba la calificación será -4
  - Primero el sistema solicita una calificación al usuario
  - Después el usuario ingresa el -4

- El -4 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el -4, es decir: Si -4 es mayor o igual a 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como -4 no es mayor o igual a 7, el algoritmo seguirá por el flujo de “No”, y enviará el mensaje “El alumno reprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.

Como se pudo observar, se realizaron cinco pruebas de escritorio usando valores que son menores, mayores o igual a 7 (que es la calificación mínima aprobatoria). Esto se debe a que siempre se deben emplear y definir valores límites (o frontera) para las pruebas de escritorio. En este caso, el 7, por estar en la condición del if, debía emplearse, así como valores menores y mayores a 7 (como fueron el 6.99 y el 7.1). Por otro lado, en el planteamiento del problema no se definieron valores límites para una calificación, es decir, cuál es el valor mínimo y máximo que puede usarse, por ello se usaron valores como 11 y -4 para las pruebas, los cuales sí son válidos para probar el algoritmo, pero no se emplean para evaluar a un alumno, al menos en México, donde en la mayoría de los casos, las calificaciones varían en el rango de 0 a 10. Esta es la razón por la que se realizaron 5 pruebas de escritorio y usando los valores antes mencionados para la variable calificación.

Ahora vayamos a visualizar el código en C#. Para la realización del código tenemos que seguir el flujo paso a paso de nuestro algoritmo, lo cual quedaría de la siguiente manera:

- Primero declaramos las variables,
- Seguido de esto, pedimos la calificación al usuario
- El usuario ingresará la calificación y ésta será comparada con nuestra condición para poder identificar el tipo de respuesta que se imprimirá al usuario
- Dependiendo del valor de la calificación, se envía un mensaje al usuario indicando si el alumno aprobó o reprobó

Bien, finalmente te mostrare la ejecución del código para dos pruebas de escritorio, una para la calificación de 6.99 y otra para la calificación de 7. Como se puede observar en las imágenes, el resultado de la ejecución del código es igual al resultado de las pruebas de escritorio.

Es importante enfatizar que cuando se plantea una condición como calificación >= 7, también existe la posibilidad de usar otra condición como calificación < 7, y ambas opciones son correctas, aunque el algoritmo sí será un poco diferente ¿podrías hacer la nueva versión del algoritmo usando la condición calificación < 7? ¿Identificas cuáles son las diferencias con respecto al que acabamos de hacer? Pausa el vídeo en esta parte y tómate un tiempo para realizarlo.

Tu algoritmo debiera parecerse a este:

(Se muestra nuevo algoritmo)

Observa que el principal cambio es que se invirtieron los resultados después de evaluar la condición del if.

Mejoremos ahora nuestro algoritmo agregando la validación para que el usuario proporcione una calificación que se encuentre en el rango de 0 – 10.
Ahora te mostraré cómo es que se construye este nuevo algoritmo:

- Paso 1: Se colocan dos swimlanes (como dijimos previamente, un swimlane es el área en la cual estarán representadas gráficamente las actividades que le corresponde hacer tanto al usuario como al sistema)
- Paso 2: Nos colocamos en el lado del swimlane del sistema para crear el inicio del diagrama, el cual está representado por un círculo negro y siempre estará en el swimlane del sistema
- Paso 3: Se requiere declarar las variables (recordemos que una variable es un espacio de memoria virtual, reservada para almacenar

un valor y es modificable a lo largo del algoritmo). Para este caso tenemos una variable llamada calificación. Esta variable (así como todas las variables) se escribe sin acento, ya que las variables no llevan acento. En este caso en concreto el tipo de variable es decimal. Esta declaración de variable se escribe en un rectángulo con los costados redondeados, el cual significa que en él debe estar escrita una actividad, y una declaración de variable es una actividad. Lo siguiente es seguir el flujo correcto de nuestro proceso a realizar, éste se describe con flechas las cuales muestran la dirección que debe seguirse.

Hagamos referencia ahora al plan de configuración, el cual habíamos dicho que es un documento formal en el que se especifican todas las configuraciones del proyecto, es decir, se estandarizan los tamaños y tipo de letra del documento, los títulos y la forma en que deben estar nombrados los diagramas e imágenes que se empleen) . Cuando se hace referencia al plan de configuración desde el diagrama de actividades, se pueden poner notas donde se coloquen comentarios, como en este caso “Declaración de variables” y éste será de color verde limón (como todas las notas) y se liga a la actividad con una línea punteada dentro del diagrama que estamos construyendo.

- Paso 4: el sistema solicita al usuario ingresar una calificación. Como se trata de una actividad, también se coloca en un rectángulo con los costados redondeados
- Paso 5: ahora, es necesario indicar que lo siguiente a realizar le corresponde al usuario, es por esta razón que el flujo de la flecha se orienta hacia la izquierda, hacia el swimlane del usuario, donde deberá ingresar una calificación. Para ello se coloca en el diagrama la variable calificación como actividad, la cual almacenará el valor ingresado. Esta actividad deberá colocarse en el swimlane del usuario, ya que es él quien proporciona la información; esto es lo único que hará el usuario. Es por esta razón que en el diagrama se visualiza que la

flecha regresa hacia el lado derecho, donde se encuentra el swimlane del sistema, ya que lo que nos falta realizar del proceso continuará allí.

- Paso 6: una vez que tenemos la calificación es necesario identificar si el valor cae en el rango de aprobada o reprobada y si es un valor válido, es decir, que sea mayor o igual a 0 y menor o igual a 10. Esto se puede hacer mediante una condición que se coloca dentro de un rombo que representa a un if.

  La condición que usaremos será:

  Si calificación >= 0 y calificación < 7 se enviará el mensaje

  “El alumno reprobó”

  Si no

  Se evaluará la calificación ingresada por el usuario en otra condición, donde:

  Si calificación >= 7 y calificación <= 10 se enviará el mensaje

  “El alumno aprobó”

  Si no

  El sistema enviará un mensaje diciendo “No ingresaste un valor válido”

  Ahora, antes de pasar a la traducción de este pseudocódigo al diagrama, es preciso mencionar que cuando se usa más de una condición, se emplean conectores. Uno de ellos es “Y”, el cual se representa en un diagrama con el signo de ampersand repetido y conecta dos condiciones. También es importante aclarar cómo se determina si una expresión que usa el conector “Y” es afirmativa o negativa: si al menos una de las condiciones que se está evaluando es negativa, toda la expresión se vuelve negativa, es decir, su resultado será no, y en el algoritmo, el flujo que deberá seguirse es el de NO. Si en la evaluación de cada una de las condiciones la respuesta es sí, y todas son sí o afirmativas, la expresión se considera afirmativa y deberá seguirse el flujo del sí.

Habiendo mencionado lo anterior, ahora sí podemos traducir el pseudocódigo al diagrama. Para esto tenemos que dentro del primer rombo escribiremos: “calificación >= 0 && calificación < 7” y tendrá dos posibles respuestas, las cuales se representarán como salidas del rombo. En caso de que la respuesta sea “SI, verdadera o afirmativa” se imprimirá una respuesta, y de ser “Negativa, falsa o un simple NO” se procederá a evaluar en un segundo rombo, en el que escribiremos: “calificación >= 7 && calificación <= 10” el cual también tendrá dos salidas de respuesta como todo rombo. Si la respuesta es “SI, verdadera o afirmativa” se imprimirá al usuario un mensaje, pero de ser “Negativa, falsa o un simple NO”, el sistema imprimirá un mensaje diciendo que no se ingresó un valor válido respecto a nuestros valores previamente delimitados.

- Paso 7: Ahora hay que colocar en el diagrama lo que sucederá cuando la respuesta es “SI” y cuando la respuesta es “NO”. Si la respuesta es “SI, Afirmativa o Verdadera” en nuestro primer rombo, deberá enviar un mensaje que indique “El alumno reprobó”. Si la respuesta es “No, Negativa o Falsa” deberá pasar a nuestra siguiente condición o segundo rombo, en donde si la respuesta es “Si, afirmativa o verdadera” se deberá enviar un mensaje que indique “El alumno aprobó”. Si la respuesta es “No, negativa o falsa” se enviará un mensaje diciendo que “No ingresaste un valor válido” tal como se mostró que se debía hacer en el paso anterior. Es importante recordar que las salidas del rombo pueden colocarse en cualquiera de sus vértices libres de otros flujos, lo cual también significa que la continuación del diagrama puede ser hacia cualquier orientación, es decir, “Arriba, Abajo, Izquierda o Derecha” dependiendo de los vértices que tenga libres.
- Paso 8: Como observamos, dependiendo del resultado de la condición, el algoritmo solo tomará un camino, el de “SI” o el de “NO”,

nunca podrá tomar los dos al mismo tiempo. Esto significa que dado el requerimiento de lo que debía hacer el algoritmo, que era definir si una calificación era aprobada o reprobada, una vez que ha evaluado todas las posibles condiciones y determinado a cuál de las dos pertenece, el diagrama deberá finalizarse, es decir ya no habrá más actividades por hacer. Por lo tanto, se deberá finalizar el diagrama para cada uno de los flujos. Un diagrama debe tener al menos un final, y en este caso los diferentes flujos apuntan hacia un final diferente, es decir, finalizan cada uno por separado.

Para finalizar el diagrama se coloca un círculo que contiene otro círculo negro en su interior.

Ahora mostraré la prueba de escritorio correspondiente a este segundo algoritmo. Para ello realizaremos 6 pruebas de escritorio:

- Para la primera prueba la calificación que usaremos será 0, que es el valor más pequeño que se definió como calificación válida de los valores frontera que se establecieron para el problema
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 0 (que es el valor con el que decidimos probar el algoritmo)
    - El 0 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 0, es decir: Si 0 es mayor o igual a 0 && 0 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como el resultado de la primera condición (0 es mayor o igual a 0) es sí o afirmativa y el resultado de la segunda condición también lo es (0 menor que 7), el resultado de ambas condiciones hacen que toda la expresión sea sí o afirmativa, por lo tanto, el algoritmo seguirá por el flujo de “Si”, y enviará el mensaje “El alumno reprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.
- Para la segunda prueba la calificación que usaremos será de 6.99

    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 6.99 (que es el valor con el que decidimos probar el algoritmo)
    - El 6.99 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 6.99, es decir: Si 6.99 es mayor o igual a 0 && 6.99 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como el resultado de la primera condición (6.99 es mayor o igual a 0) es sí o afirmativa, y el resultado de la segunda condición también lo es (6.99 menor que 7), el resultado de ambas condiciones hacen que toda la expresión sea sí o afirmativa, por lo tanto, el algoritmo seguirá por el flujo de “Si”, y enviará el mensaje “El alumno reprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.
- Para la tercera prueba la calificación será de 7
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 7 (que es el valor con el que decidimos probar el algoritmo)
    - El 7 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 7, es decir: Si 7 es mayor o igual a 0 && 7 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como el resultado de la primera condición (7 es mayor o igual a 0) es sí o afirmativa, pero el resultado de la segunda condición (7 menor que 7) es negativa o no, entonces toda la expresión se convierte en negativa, por lo que seguirá el algoritmo por el flujo de “No”. Continuando sobre esa dirección, nos encontramos que se requiere evaluar otro par de condiciones: si 7 es mayor o igual a 7, lo cual es afirmativo, y si 7 es menor o igual a 10, lo cual también es cierto. Esto nos convierte toda la expresión en positiva y el algoritmo seguirá por el flujo de “Sí” enviando el mensaje “El alumno aprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.

- Para la cuarta prueba la calificación será de 10
    - Primero el sistema solicita una calificación al usuario (para este caso es 10)
    - Después el usuario ingresa el 10 (que es el valor con el que decidimos probar el algoritmo)
    - El 10 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 10, es decir: Si 10 es mayor o igual a 0 && 10 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como el resultado de la primera condición (10 es mayor o igual a 0) es sí o afirmativa, pero el resultado de la segunda condición (10 es menor que 7) es negativa o no, entonces toda la expresión se convierte en negativa, por lo que seguirá el algoritmo por el flujo de “No”. Continuando sobre esa dirección, nos encontramos que se requiere evaluar otro par de condiciones: si 10 es mayor o igual a 7, lo cual es afirmativo, y si 10 es menor o igual que 10, lo cual también es cierto. Esto nos convierte toda la expresión en positiva y el algoritmo seguirá por el flujo de “Sí” enviando el mensaje “El alumno aprobó”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.
- Para la quinta prueba la calificación será -1
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el -1 (que es el valor con el que decidimos probar el algoritmo)
    - El -1 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el -1, es decir: Si -1 es mayor o igual a 0 && -1 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como -1 no es mayor o igual a 0 y sí es menor que 7, entonces toda la expresión se convierte en negativa, por lo que seguirá el algoritmo por el flujo de “No”. Continuando sobre esa dirección, nos encontramos que se requiere evaluar otro par de condiciones: si -1 es mayor o igual a 7, lo cual es negativo o no, y si -

1 es menor o igual a 10, lo cual sí es cierto. En este caso, como uno de los resultados es negativo y el otro positivo, entonces toda la expresión se convierte en negativa, por lo que el algoritmo seguirá por el flujo de “No” enviando el mensaje “No ingresaste un valor válido”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.

- Para la sexta prueba la calificación será 11
    - Primero el sistema solicita una calificación al usuario
    - Después el usuario ingresa el 11 (que es el valor con el que decidimos probar el algoritmo)
    - El 11 será evaluado en el if. Esto significa que se sustituirá la variable calificación por el 11, es decir: Si 11 es mayor o igual a 0 && 11 es menor que 7 entonces continuar hacia el flujo de “si” o de “no”. En este caso, como 11 es mayor o igual a 0 y no es menor que 7, el algoritmo seguirá por el flujo de “No”, ya que deben ser afirmativas las dos condiciones para poder ser cierto. Continuando sobre esa dirección, nos encontramos que se requiere evaluar otro par de condiciones: si 11 es mayor o igual a 7, lo cual es afirmativo o sí, y si 11 es menor o igual a 10, lo cual es negativo o no. En este caso, como uno de los resultados es negativo y el otro positivo, entonces toda la expresión se convierte en negativa, por lo que el algoritmo seguirá por el flujo de “No” enviando el mensaje “No ingresaste un valor válido”. Posteriormente, finaliza el algoritmo y por lo tanto, nuestra prueba de escritorio.

Como se pudo observar, se realizaron seis pruebas de escritorio usando valores que son menores, mayores o igual a 7 (que es la calificación mínima aprobatoria), de las cuales en dos de ellas se emplearon los valores frontera contra los cuales se requería validar el rango de calificaciones que podía tener un alumno (esto es 0 y 10); en otras 2 pruebas se usaron valores que estaban fuera del rango de calificaciones definido (como el -1 y el 11), y los dos valores restantes

correspondieron al 7, que era la calificación mínima aprobatoria que se usaría para la condición, y el 6.99, que representaba a una calificación menor a 7.
Con estos 6 valores pudimos agotar todas las posibles opciones que puede tomar la variable calificación, y dado que en todas se obtuvo un resultado satisfactorio, se puede concluir que nuestro algoritmo está correcto.
Ahora vayamos a visualizar el código en C#. Para la realización del código tenemos que seguir el flujo paso a paso de nuestro algoritmo, lo cual quedaría de la siguiente manera:

- Primero declaramos las variables,
- Seguido a esto, pedimos la calificación al usuario
- El usuario ingresará la calificación y ésta será comparada con nuestras condiciones para poder identificar el tipo de respuesta que se imprimirá al usuario
- Dependiendo del valor de la calificación, se envía un mensaje al usuario indicando si el alumno aprobó, reprobó o si se ingresó un valor que no es válido.

Finalmente te mostrare la ejecución del código para tres pruebas de escritorio, una para la calificación de 6.99, otra para la calificación de 7 y una más para la calificación -1. Como se puede observar en las imágenes, el resultado de la ejecución del código es igual al resultado de las pruebas de escritorio.
Bien, hasta aquí el segundo video, recuerda que si tienes dudas, puedes ver este video nuevamente o regresar y pausar donde no haya quedado claro, nos vemos en el siguiente video.

3.6 Video 3. Switch

En el tercer video de este material, se abordó a la estructura de control switch, en el cual se utilizó un ejemplo diferente pero siguiendo el mismo orden. Ahora, se ingresará una calificación entre 0 y 10, y dependiendo cuál haya sido el valor ingresado se enviará un mensaje a manera representativa del valor de ésta.

Cabe mencionar, que en este diagrama se anexaron notas de color blanco, con la finalidad de ser mas explícitos con los elementos que contiene el switch y que lo diferencian del if.

Durante la redacción del script se explica detalladamente este diagrama:

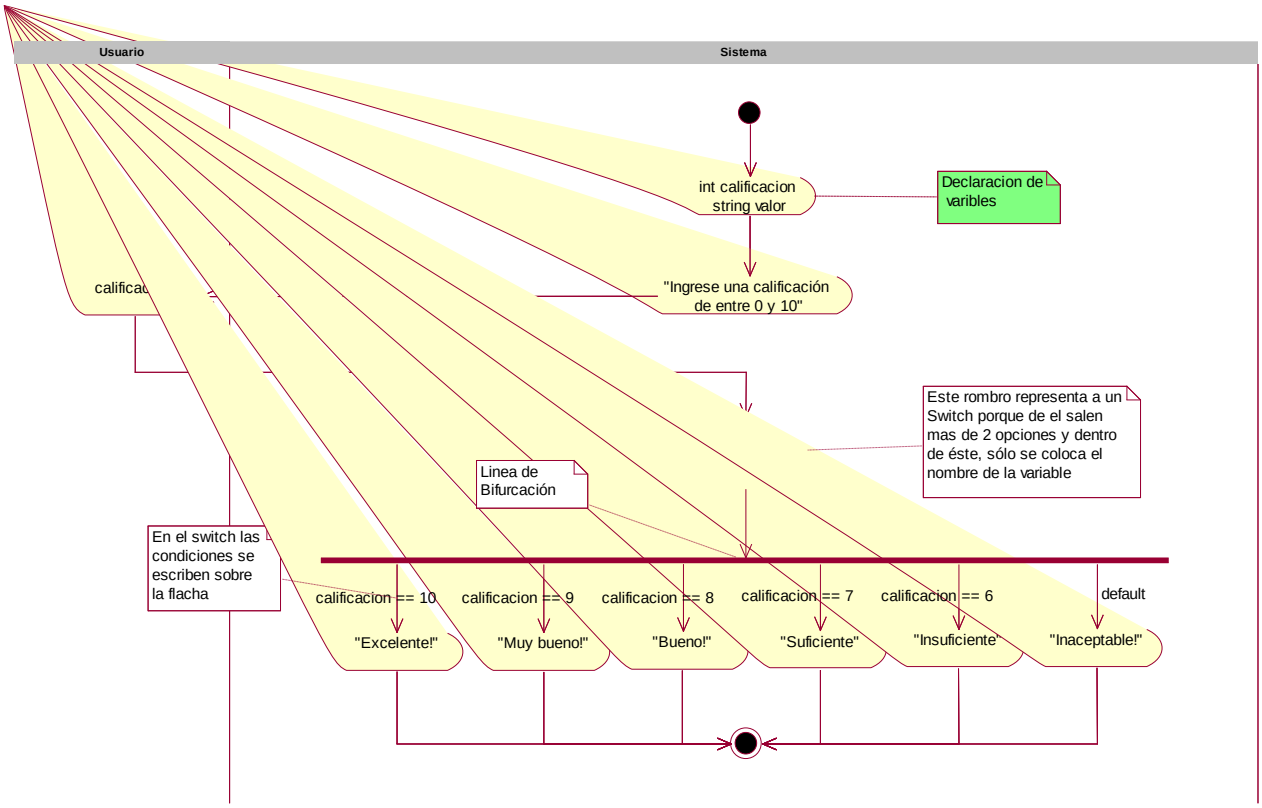

Figura 6. Diagrama de Actividades Switch

3.6.1 Prueba de escritorio

Para la evaluación del algoritmo anterior, se realizaron seis pruebas de escritorio, cada prueba por cada caso contenido en el algoritmo.

La explicación detallada de esta prueba de escritorio con el algoritmo se muestra en el video y en el script referente a esta estructura de control.

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	10	Excelente
2	9	Muy Bueno
3	8	Bueno
4	7	Suficiente
5	6	Insuficiente
6	3	Inaceptable

Tabla 12. Prueba de escritorio para el switch

3.6.2 Código en C#

Ahora se visualiza el código correspondiente a esta nueva estructura de control, el cual fue diseñado con base en el algoritmo. También se describe de manera muy puntual en el script correspondiente a esta estructura de control.

```
static void Main(string[] args)
        {
            //Declaración de Variables
            int calificacion;
            string valor;

            //Solicita calificación al usuario
            Console.WriteLine("Ingrese una calificación de entre 0 y 10: ");
            valor = Console.ReadLine();
            calificacion = Convert.ToInt32(valor);

            //Inicio del Switch
            switch (calificacion)
            {
                //Caso 1
                case 10:
                    Console.WriteLine("Excelente!");
                    break;
                //Caso 2
```

```
            case 9:
                Console.WriteLine("Muy bueno!");
                break;
            //Caso 3
            case 8:
                Console.WriteLine("Bueno!");
                break;
            //Caso 4
            case 7:
                Console.WriteLine("Suficiente");
                break;
            //Caso 5
            case 6:
                Console.WriteLine("Insuficiente");
                break;
            //Caso 6
            default:
                Console.WriteLine("Inaceptable!");
                break;
    }
}
```

3.6.3 Resultado de la ejecución del código en C#

Finalmente se visualiza la ejecución del código para cada uno de los casos correspondientes.

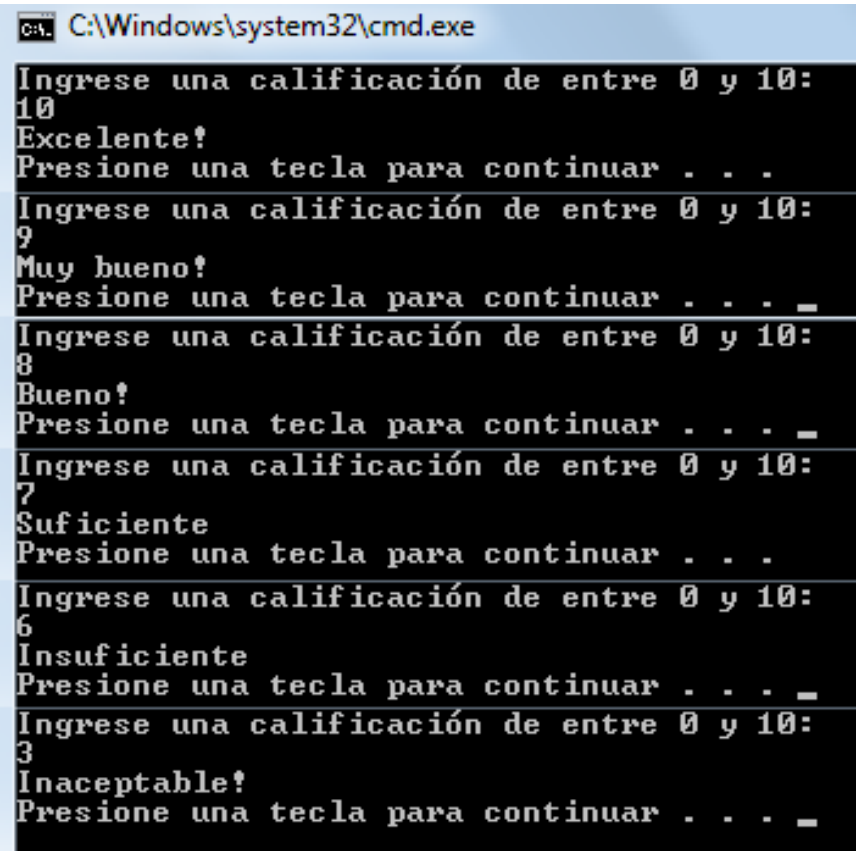

Figura 7. Resultado de la Ejecución del Código Switch

3.6.4 Script

En este script se describe de manera muy puntual en qué consiste esta nueva estructura de control y cómo funciona basándose en un ejemplo bastante sencillo. Además de explicar detalladamente los componentes de cada uno de los elementos que lo integran, se muestra el funcionamiento del algoritmo con las pruebas de escritorio, la elaboración del código en C# y finalmente su correspondiente ejecución de código.

A continuación se presenta el script del video:

Hola mi nombre es Román, estudié la carrera de Ing. de Software en la Universidad Madero, nuevamente estoy contigo para esta serie de videos.

Es importante recordarte, que si tienes dudas respecto de algún término, regreses y visualices el primer video y el segundo video en donde se explican a detalle los términos y conceptos de lo visto durante estos videos, además de explicarte los componentes y significados de las diferentes figuras que se emplean en los diagramas.

De estar todo claro, continuemos con este video en el cual abordaremos la estructura de control Switch. Esta estructura de control funciona de manera muy similar al if, ya que consiste en una toma de decisión de acuerdo a la condición que previamente se haya planteado. Ahora no tendremos solo dos opciones de salida como lo son un “si” y un “no”, tendremos más de 3 caminos los cuales tomar. Para ejemplificar de mejor manera esto, te mostraré el siguiente problema, el cual está basado en el problema anterior del IF, en cuanto a calificaciones:

Problema 1: Realizar un algoritmo gráfico en forma de diagrama de actividades que solicite al usuario una calificación y en base a su respuesta, muestre a manera de mensaje el desempeño del alumno con los siguientes mensajes: “Excelente!” para un 10, “Muy bueno!” para un 9, “Bueno!” para un 8, “Suficiente” para un 7, “Insuficiente” para un 6 y finalmente un “Inaceptable!” para el resto de calificaciones.

Bueno, como ya se mostró en el video anterior, lo primero a realizar en el diagrama de actividades es:

- Dibujar dos swimlanes, uno para representar lo que el usuario realizaría y otro para indicar lo que el sistema elaborará.
- Lo siguiente es dibujar un punto negro del lado del swimlane del sistema para indicar el comienzo del algoritmo.
- Ahora, declaramos las variables necesarias para el correcto funcionamiento de nuestro programa.
- Después, el sistema pedirá al usuario ingresar una calificación.
- Ahora le corresponde al usuario ingresar la calificación; es por esta razón que la dirección de la flecha se dirige hacia el swimlane del usuario, al lado izquierdo de tu pantalla.

- Al ser ingresado el valor por el usuario, se almacena en nuestra variable “calificación”, regresando hacia el sentido del swimlane del sistema. Donde, este rombo, representa a un switch, porque de él salen más de dos opciones y dentro de este, solo se coloca el nombre de la variable.
- Ahora, para mejor entendimiento y diseño del diagrama se emplea una barra gruesa para mostrar de mejor manera las diferentes opciones que puede tomar el diagrama. A esta barra se le conoce como “Línea de Bifurcación”. Dependiendo del diseño del diagrama, esta línea puede estar dibujada de manera horizontal o vertical.
- Es importante mencionar que las condiciones en el switch se escriben sobre las flechas y no como una actividad. A este conjunto de condiciones, se les conoce como “casos” y, dependiendo de qué caso se cumpla, mostrará un resultado. También se tiene un caso más de “default”, este funciona cuando todos los casos anteriores fueron falsos.
- Finalmente, terminará nuestro algoritmo.

Bien, es momento de realizar las pruebas necesarias, y para esto utilizaremos los valores que se muestran en la siguiente tabla. Para este algoritmo, realizaremos 6 pruebas de escritorio.

- Para la primera utilizaremos una calificación de 10
- El sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para este caso es “10”, ahora “10” representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “10”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 10 es igual a 10, como esto es verdadero, lo siguiente a realizar es que se imprime al usuario el mensaje de “Excelente!”.
- Finalmente terminaría nuestro algoritmo.

Como pudimos notar, el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

- Para la segunda prueba utilizaremos la calificación de 9
- Primero el sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para esta prueba será el 9, ahora 9 representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “9”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 9 es igual a 10, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 9 es igual a 9, como esto es verdadero, lo siguiente a realizar es que se imprime al usuario el mensaje de “Muy Bueno!”.
- Finalmente terminaría nuestro algoritmo.

Nuevamente el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

- Para la tercera prueba utilizaremos la calificación de 8
- Primero el sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para esta prueba será el 8, ahora 8 representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “8”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 8 es igual a 10, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 8 es igual a 9, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 8 es igual a 8, como esto es verdadero, lo siguiente a realizar es que se imprime al usuario el mensaje de “Bueno!”.
- Finalmente terminaría nuestro algoritmo.

Nuevamente el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

- Para la cuarta prueba utilizaremos la calificación de 7

- Primero el sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para esta prueba será el 7, ahora 7 representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “7”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 7 es igual a 10, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 7 es igual a 9, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 7 es igual a 8, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 7 es igual a 7, como esto es verdadero, lo siguiente a realizar es que se imprime al usuario el mensaje de “Suficiente”.
- Finalmente terminaría nuestro algoritmo.

Nuevamente el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

- Para la quinta prueba utilizaremos la calificación de 6
- Primero el sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para esta prueba será el 6, ahora 6 representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “6”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 6 es igual a 10, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 6 es igual a 9, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 6 es igual a 8, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 6 es igual a 7, como esto es falso, se pasa a evaluar en el siguiente caso.

    - Donde 6 es igual a 6, como esto es verdadero, lo siguiente a realizar es que se imprime al usuario el mensaje de “Insuficiente”.
- Finalmente terminaría nuestro algoritmo.

Nuevamente el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

- Para la sexta prueba utilizaremos la calificación de 3
- Primero el sistema pedirá al usuario ingresar una calificación
- Después el usuario ingresará la calificación, que para esta prueba será el 3, ahora 3 representa a nuestra variable calificación.
- En el switch, la variable calificación (que ahora es “3”), pasará a ser evaluada en los diferentes casos. En el primero se tiene que:
    - 3 es igual a 10, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 3 es igual a 9, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 3 es igual a 8, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 3 es igual a 7, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde 3 es igual a 6, como esto es falso, se pasa a evaluar en el siguiente caso.
    - Donde, como todos los casos anteriores resultaron ser falsos, se toma el caso de “default” en donde se imprime al usuario el mensaje de “Inaceptable!”.
- Finalmente terminaría nuestro algoritmo.

Nuevamente el resultado de esta prueba de escritorio resultó ser igual al del algoritmo, por lo tanto fue una prueba correcta, así como nuestro algoritmo.

Ahora visualicemos el código de este diagrama, que basado en el algoritmo, quedaría de esta manera:

- Primero tenemos la declaración de Variables
- Después solicitamos una calificación al usuario
- Lo siguiente es el inicio del switch
- Después se tienen los diferentes casos con sus respectivos mensajes

Finalmente te mostraré la ejecución del código por cada una de las pruebas de escritorio:

- Esta es la ejecución del código para la primera prueba, como se observa el resultado es el mismo
- Ahora tenemos la ejecución del código para la segunda prueba, como se observa el resultado es el mismo
- Después la ejecución del código para la tercera prueba, como se observa el resultado es el mismo
- Esta es la ejecución del código para la cuarta prueba, como se observa el resultado es el mismo
- Ahora tenemos la ejecución del código para la quinta prueba, como se observa el resultado es el mismo
- Y finalmente la ejecución del código para la sexta prueba, como se observa el resultado es el esperado.

Bueno, hasta aquí este tercer video, en el cual vimos el concepto y funcionamiento del switch. Espero que haya sido de gran ayuda para ti. Nos vemos en el siguiente video, para conocer una estructura de control más.

3.7 Video 4. For

Para esta nueva estructura de control, se realizaron varios tipos de diagramas, cada uno de ellos mostrando el mismo ejemplo pero con variaciones en el algoritmo que al final repercutirían tanto en el código, como en la ejecución del mismo. Se optó por no cambiar de ejemplo con la finalidad de mostrar los cambios que se tendrían

al modificar ciertos valores con los que se caracteriza la estructura de control “for” y así ser más evidente con el funcionamiento y principales cambios que se tendrían.

La prueba de escritorio para todos los diagramas fue la misma, como se mencionó con anterioridad, sin embargo, el código sí sufre ligeras modificaciones dependiendo del algoritmo que se visualice en su momento. Esto, arroja como resultado que en la ejecución del código se visualicen resultados diferentes.

Finalmente se cuenta con un script para este video, el cual, narra de manera detallada cada uno de los elementos de los diagramas, la correcta elaboración de la prueba de escritorio, la creación del código basándose en el algoritmo diseñado, y culminando con el resultado de la ejecución del código.

Cabe mencionar, que en los diagramas se anexaron notas de color blanco, con la finalidad de enfatizar sobre ciertos componentes de esta estructura de control, pero en realidad, cuando se realiza el diagrama no tiene por qué llevarlos.

A continuación se listan cada uno de los elementos para la creación de este cuarto video llamado For:

3.7.1 For simple

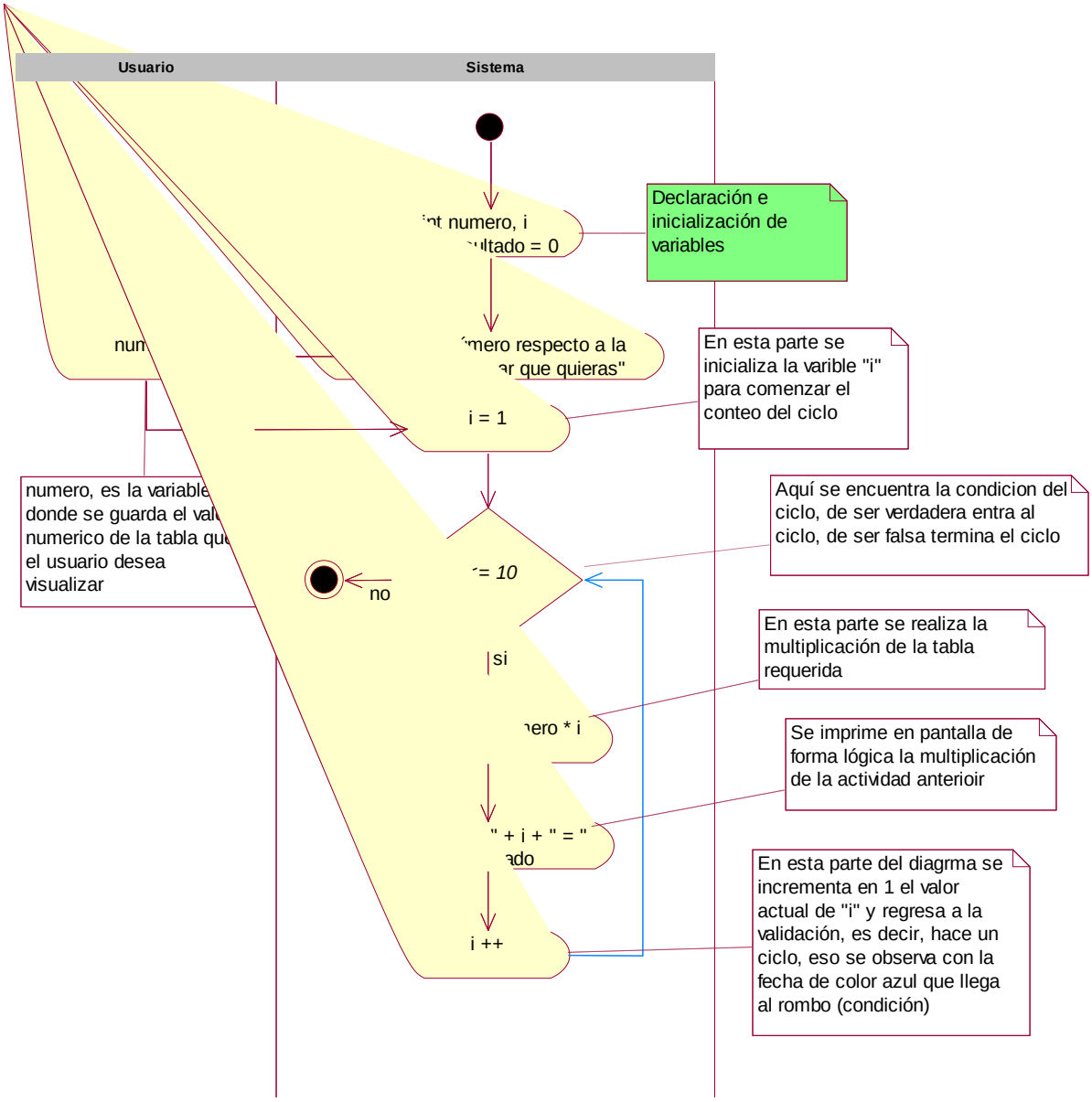

Figura 8. Diagrama de actividades for simple

3.7.2 Prueba de Escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	2	Tabla de multiplicar

Tabla 13. Prueba de escritorio for simple

3.7.3 Código en C#

```
static void Main(string[] args)
        {
            //Declaración e inicialización de Variables
            int numero;
            int resultado = 0;

            //Solicitamos al usuario ingresar un número
            Console.WriteLine("Ingresa un número respecto a la tabla de multiplicar
que quieras:");
            numero = int.Parse(Console.ReadLine());

            /*En esta parte se inicializa la varible "i" para comenzar el conteo del
ciclo
             * tambien se encuentra la condición del ciclo donde de ser verdadera
entra al ciclo
             * y de ser falsa termina el ciclo
             * también se incrementa en 1 el valor actual de "i" y regresa a la
validación
             * es decir, hace un ciclo*/
            for (int i = 1; i <= 10; i++)
            {
                //En esta parte se realiza la multiplicación de la tabla requerida
                resultado = numero * i;
                //Se imprime en pantalla la forma lógica la multiplicación
                Console.WriteLine(numero+ " * " +i+ " = " +resultado);
            }
        }
```

3.7.4 Resultado de la ejecución del código en C#

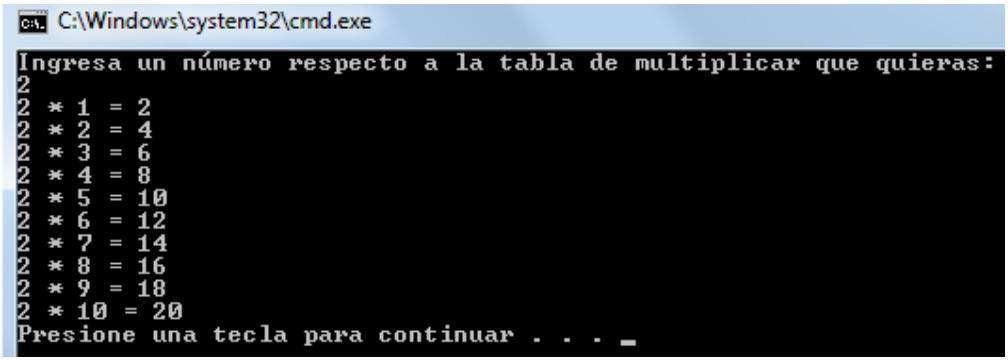

Figura 9. Resultado de la ejecución del código for simple

3.7.5 For ascendente en números pares

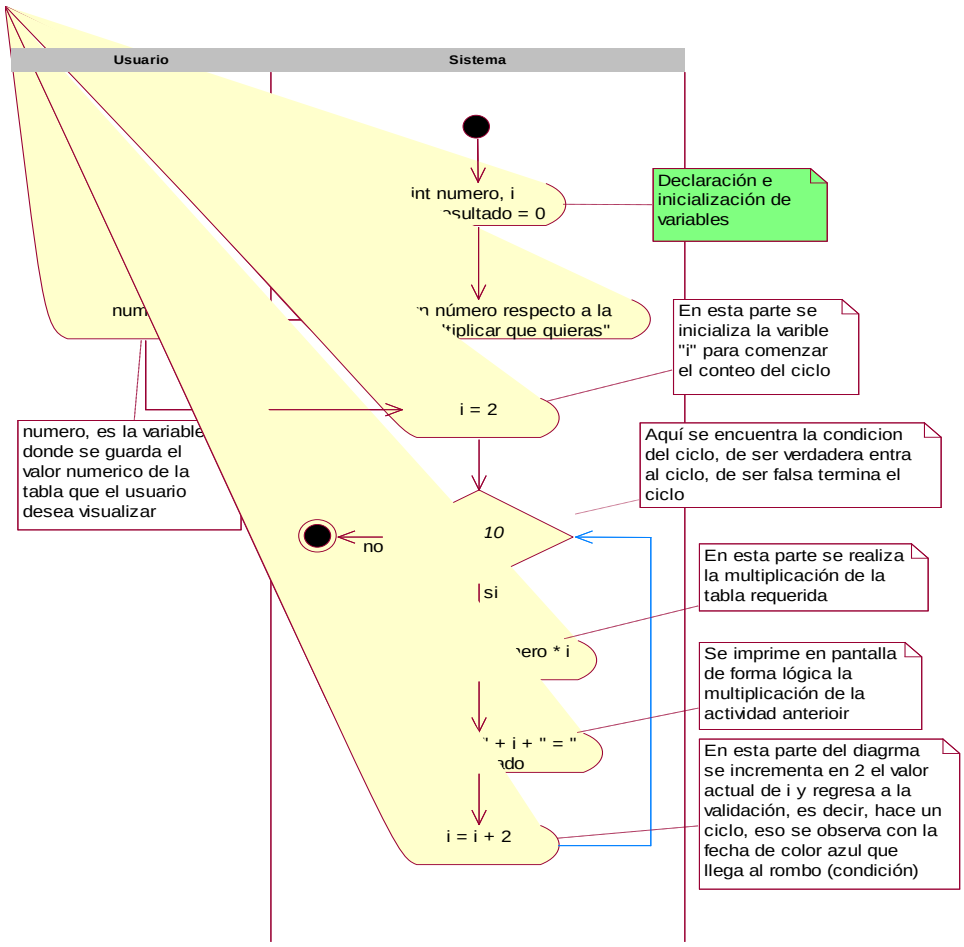

Figura 10. Diagrama de actividades for ascendente en números pares

3.7.6 Prueba de escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	2	Tabla de multiplicar

Tabla 14. Prueba de escritorio for ascendente en números pares

3.7.7 Código en C#

```
static void Main(string[] args)
        {
            //Declaración e inicialización de Variables
            int numero, resultado;

            //Solicitamos al usuario ingresar un número
            Console.WriteLine("Ingresa un número respecto a la tabla de multiplicar
que quieras:");
            numero = int.Parse(Console.ReadLine());

            /*En esta parte se inicializa la varible "i" para comenzar el conteo del
ciclo
            * tambien se encuentra la condición del ciclo donde de ser verdadera
entra al ciclo
            * y de ser falsa termina el ciclo
            * también se incrementa en 2 el valor actual de "i" y regresa a la
validación
            * es decir, hace un ciclo*/
            for (int i = 2; i <= 10; i = i + 2)
            {
                //En esta parte se realiza la multiplicación de la tabla requerida
                resultado = numero * i;
                //Se imprime en pantalla la forma lógica la multiplicación
                Console.WriteLine(numero + " * " + i + " = " + resultado);
            }
        }
```

3.7.8 Resultado de la ejecución del código en C#

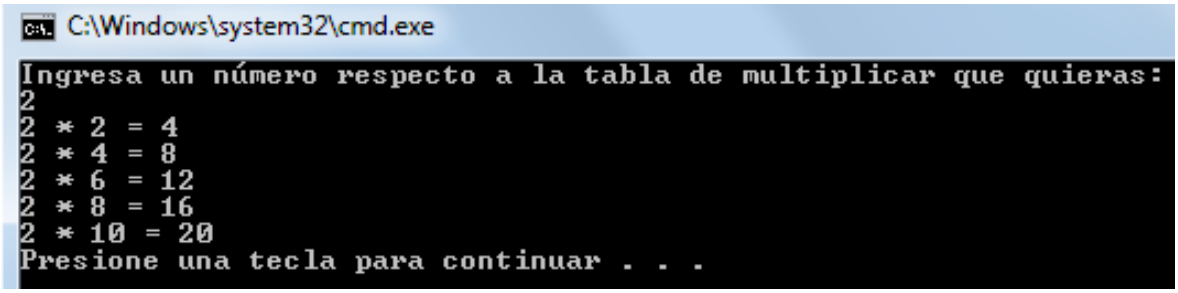

Figura 11. Resultado de la ejecución del código for ascendente en números pares

3.7.9 For ascendente con números impares

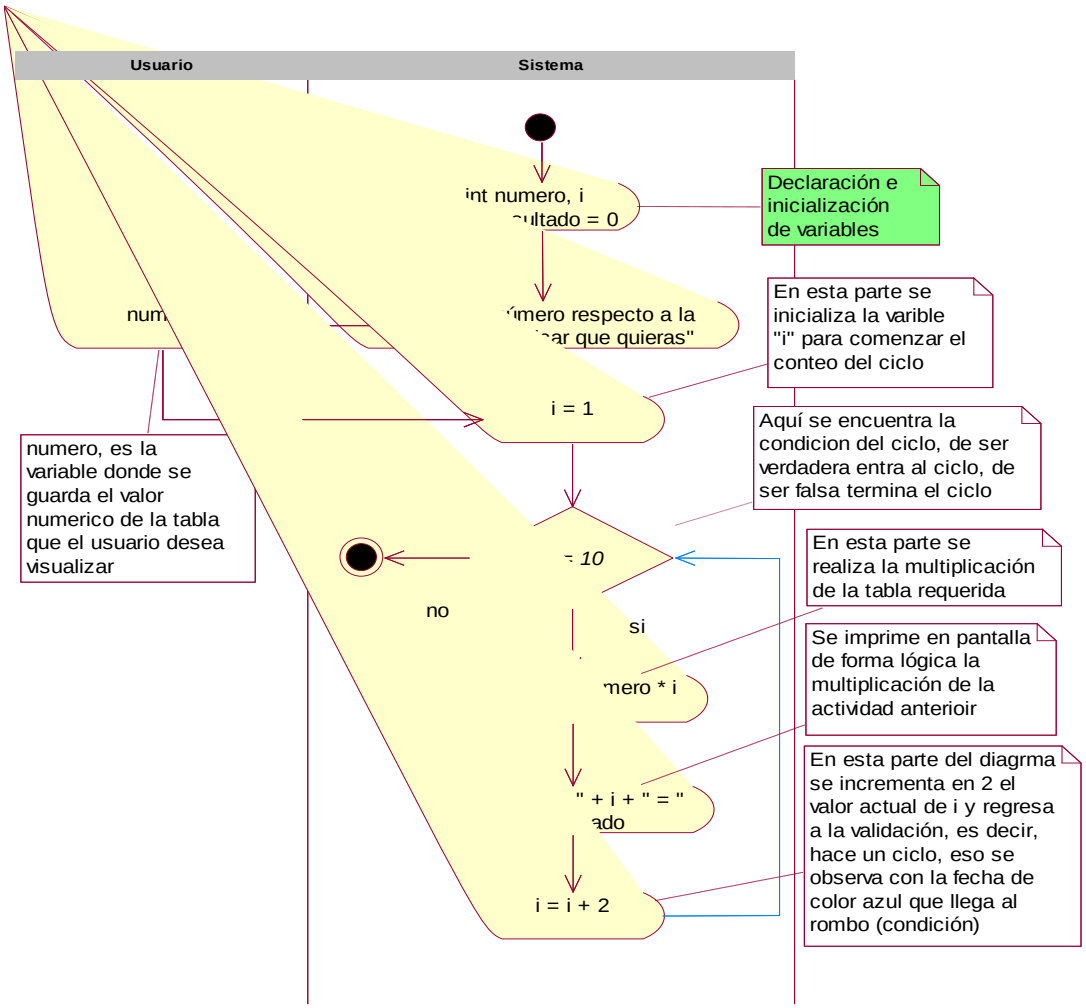

Figura 12. Diagrama de actividades for ascendente con números impares

3.7.10 Prueba de escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	2	Tabla de multiplicar

Tabla 15. Prueba de escritorio for ascendente con números impares

3.7.11 Código en C#

```
static void Main(string[] args)
        {
            //Declaración e inicialización de Variables
            int numero, resultado;

            //Solicitamos al usuario ingresar un número
            Console.WriteLine("Ingresa un número respecto a la tabla de multiplicar
que quieras:");
            numero = int.Parse(Console.ReadLine());

            /*En esta parte se inicializa la varible "i" para comenzar el conteo del
ciclo
            * tambien se encuentra la condición del ciclo donde de ser verdadera
entra al ciclo
            * y de ser falsa termina el ciclo
            * también se incrementa en 2 el valor actual de "i" y regresa a la
validación
            * es decir, hace un ciclo*/
            for (int i = 1; i <= 10; i = i + 2)
            {
                //En esta parte se realiza la multiplicación de la tabla requerida
                resultado = numero * i;
                //Se imprime en pantalla la forma lógica la multiplicación
                Console.WriteLine(numero + " * " + i + " = " + resultado);
            }
        }
```

3.7.12 Resultado de la ejecución del código en C#

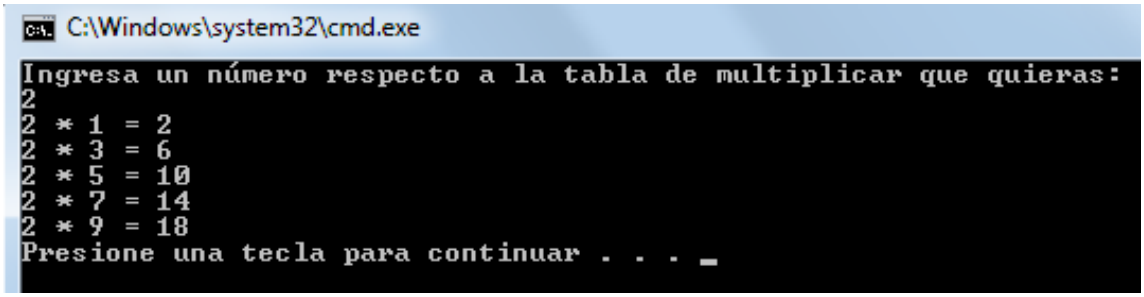

Figura 13. Resultado de la ejecución del código for ascendente con números impares

3.7.13 For Descendente

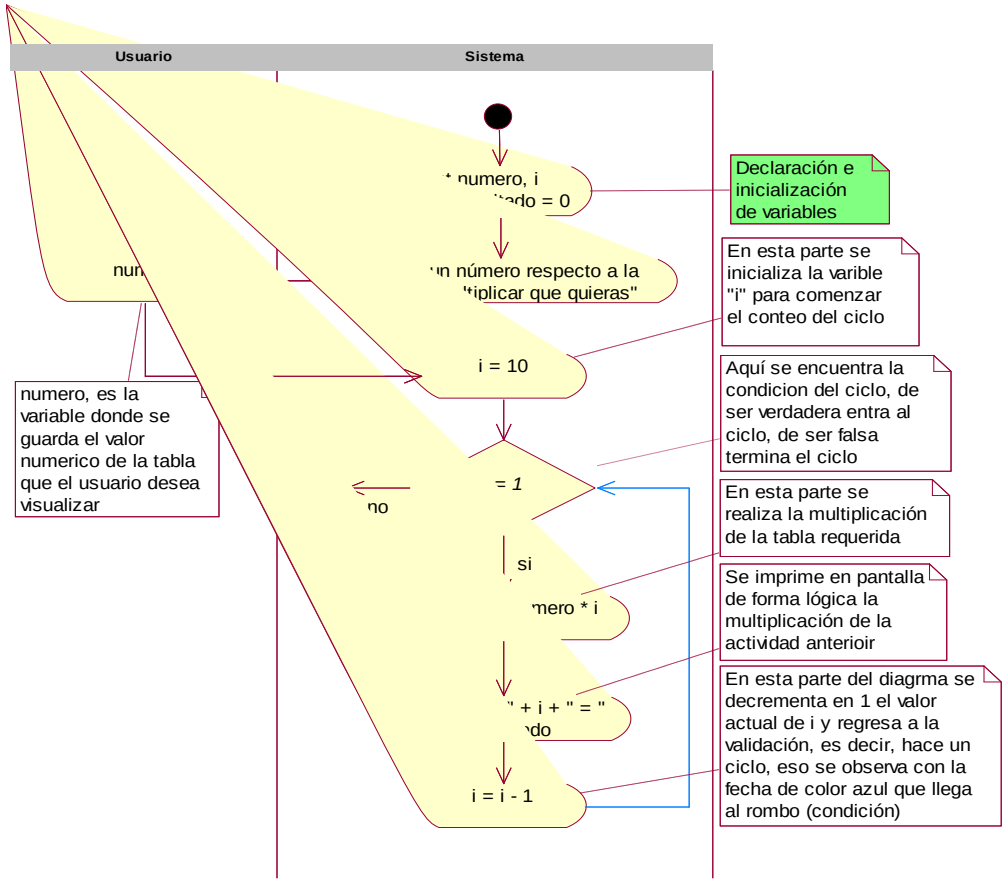

Figura 14. Diagrama de actividades for descendente

3.7.14 Prueba de escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	2	Tabla de multiplicar

Tabla 16. Prueba de escritorio for descendente

3.7.15 Código en C#

```
static void Main(string[] args)
        {
            //Declaración e inicialización de Variables
            int numero, resultado;

            //Solicitamos al usuario ingresar un número
            Console.WriteLine("Ingresa un número respecto a la tabla de multiplicar
que quieras:");
            numero = int.Parse(Console.ReadLine());

            /*En esta parte se inicializa la varible "i" para comenzar el conteo del
ciclo
            * tambien se encuentra la condición del ciclo donde de ser verdadera
entra al ciclo
            * y de ser falsa termina el ciclo
            * también se decrementa en 1 el valor actual de "i" y regresa a la
validación
            * es decir, hace un ciclo*/
            for (int i = 10; i >= 1; i = i - 1)
            {
                //En esta parte se realiza la multiplicación de la tabla requerida
                resultado = numero * i;
                //Se imprime en pantalla la forma lógica la multiplicación
                Console.WriteLine(numero + " * " + i + " = " + resultado);
            }
        }
```

3.7.16 Resultado de la ejecución del código en C#

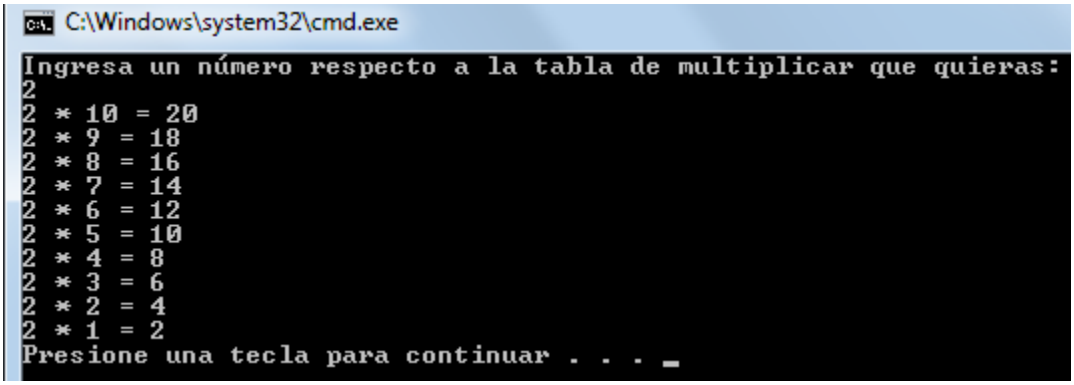

Figura 15. Resultado de la ejecución del código for descendente

3.7.17 For descendente con números pares

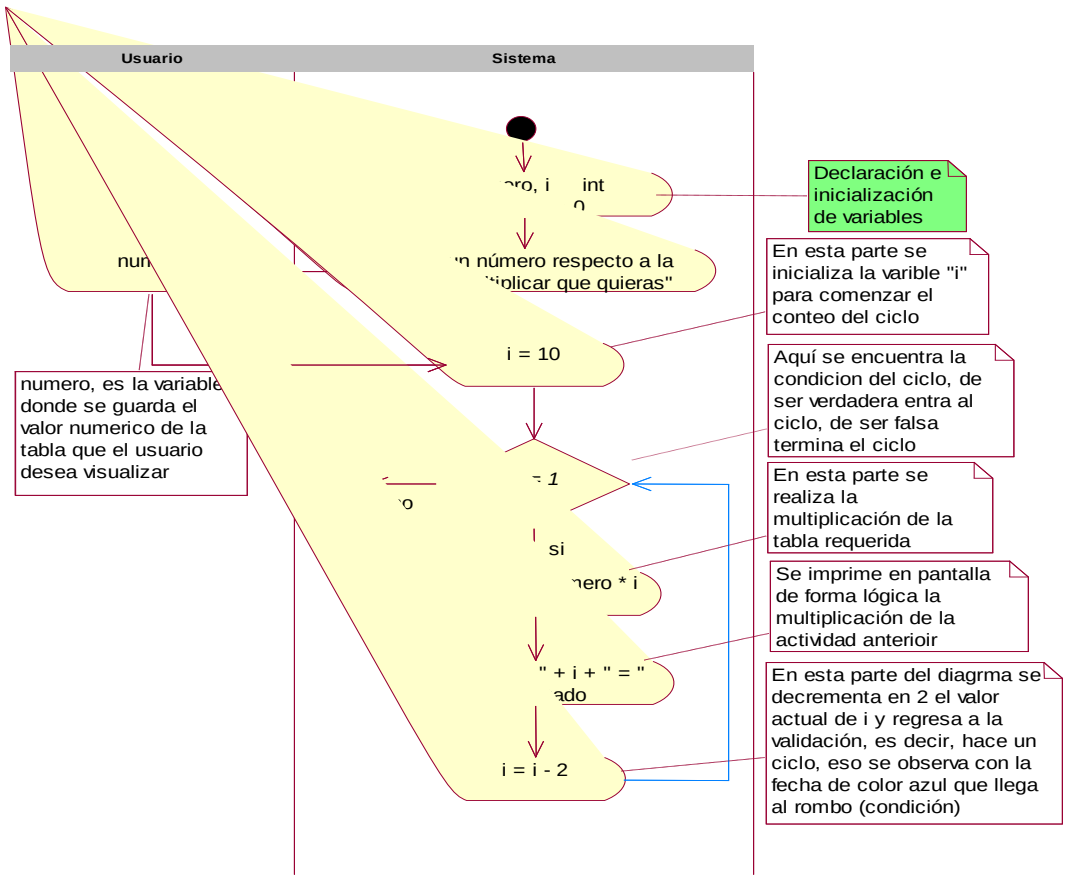

Figura 16. Diagrama de actividades for descendente con números pares

3.7.18 Prueba de escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	2	Tabla de multiplicar

Tabla 17. Prueba de escritorio for descendente con números pares

3.7.19 Código en C#

```
static void Main(string[] args)
        {
            //Declaración e inicialización de Variables
            int numero, resultado;

            //Solicitamos al usuario ingresar un número
            Console.WriteLine("Ingresa un número respecto a la tabla de multiplicar
que quieras:");
            numero = int.Parse(Console.ReadLine());

            /*En esta parte se inicializa la varible "i" para comenzar el conteo del
ciclo
            * tambien se encuentra la condición del ciclo donde de ser verdadera
entra al ciclo
            * y de ser falsa termina el ciclo
            * también se decrementa en 2 el valor actual de "i" y regresa a la
validación
            * es decir, hace un ciclo*/
            for (int i = 10; i >= 1; i = i - 2)
            {
                //En esta parte se realiza la multiplicación de la tabla requerida
                resultado = numero * i;
                //Se imprime en pantalla la forma lógica la multiplicación
                Console.WriteLine(numero + " * " + i + " = " + resultado);
            }
        }
```

3.7.20 Resultado de la ejecución del Código en C#

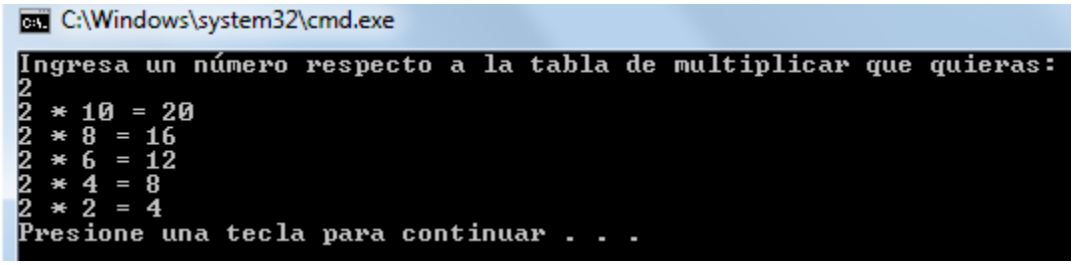

Figura 17. Resultado de la ejecución del código for descendente con números pares

3.7.21 Script

Hola mi nombre es Román, estudie la carrera de Ingeniería de Software en la Universidad Madero. En esta ocasión te explicaré en qué consiste y cómo funciona el ciclo “for”. Antes de comenzar es importante mencionar que si no tienes claro algún concepto, es necesario que revises nuevamente el primer y segundo video de este material, ya que podría causarte problemas algún término explicado en este video. De estar todo bien, te invito a continuar.

El ciclo for es una estructura de control la cual nos permite ejecutar de manera repetitiva un bloque de instrucciones. Ahora pasemos al ejemplo para que esto te quede mucho más claro.

Problema: Realizar un algoritmo a manera de diagrama de actividades que pida al usuario ingresar un número y muestre la tabla de multiplicar de este número.

Como ya hemos visto anteriormente, lo primero que se debe hacer es:

- Trazar dos swimlanes, uno para representar lo que realizará el usuario y otro para indicar lo que realizará el sistema.
- Seguido de esto indicamos el inicio del algoritmo con un círculo negro.
- Después, se deben declarar las variables necesarias para nuestro algoritmo.

- La siguiente actividad es que el sistema solicitará ingresar un número en representación de una tabla de multiplicar que el usuario desee conocer.
- A lo que el usuario ingresa un número. Número, es la variable donde se guarda el valor numérico de la tabla que el usuario desea visualizar
- En la siguiente actividad se inicializa la variable “i”, para comenzar el conteo del ciclo; éste iniciará en 1.
- En el rombo se declara la condición del ciclo, donde, de ser verdadera, entra al ciclo, y de ser falsa, termina el ciclo y nuestro algoritmo. Con esta condición podremos indicar cuántas veces se repetirá nuestro ciclo. Así que tenemos que mientras “i” sea menor o igual que 10 este se estará repitiendo.
- Si la condición es verdadera, se continuará por el camino del “si” y se ejecuta una actividad en donde se realiza la multiplicación de la tabla requerida, donde el resultado será igual al número que el usuario ingresó que corresponde a la tabla, multiplicado por el valor actual de la variable “i” que declaramos al inicio
- Lo siguiente es imprimir el resultado para ser visualizado en pantalla; en esta parte se imprime de forma lógica la multiplicación de la actividad anterior, ésta, quedará mejor entendida cuando se muestre el código.
- Al final de esta parte del diagrama, se incrementa en 1 el valor actual de "i" y regresa a la validación, es decir, hace un ciclo; eso se observa con la fecha de color azul que llega al rombo en donde se encuentra nuestra condición.

Cabe mencionar que para este diagrama se anexaron comentarios de color blanco únicamente para mayor entendimiento de este algoritmo, pero en realidad no tiene por qué llevarlos, únicamente debe llevar el comentario de color verde.

Realicemos nuestra prueba de escritorio para corroborar que nuestro algoritmo funciona adecuadamente; para esta prueba tomaremos el valor de 2, es decir, queremos conocer la tabla del número 2

Así que lo primero es que:

- El sistema solicita ingresar un número

- Ingresamos un número, para esta prueba, el número 2; ahora 2 representa a la variable “número”
- Lo siguiente es que se inicializa la variable “i” con el valor 1
- Después se evalúa la condición, donde en su primer ciclo tenemos que:
    - “i”, que ahora tiene el valor de 1, es menor o igual a 10
    - como esto es cierto continuamos por el camino del “si”
- Así que el resultado es igual a 2 por 1, lo cual es 2
- Posteriormente lo que se hace es imprimir el resultado de esta primera multiplicación
- Finalmente se incrementa en 1 la variable “i”, es decir ahora “i” tiene el valor de 2 y ya no 1 y regresa a la condición, donde:
- La variable “i” será evaluada nuevamente, entonces:
- 2 es menor o igual que 10, como esto es cierto, continuamos por el camino del “si”
- Dónde lo siguiente a realizar es la multiplicación, así que tenemos que resultado es igual a 2 * 2, lo cual es 4
- Después procedemos a realizar la impresión de esta segunda operación
- Finalmente se incrementa nuevamente la variable “i” en uno, ahora “i” tiene el valor de 3 y será evaluada nuevamente en nuestro ciclo.

Como podemos observar en nuestra prueba de escritorio, la variable “i” se estará incrementando en uno; hasta el momento tenemos que la variable “i” tiene el valor de 3, lo cual nos lleva a concluir que siempre se estará incrementando de uno en uno hasta llegar al 10, lo cual tiene como resultado que la tabla del 2 se imprimirá por completo, pero el ciclo aun no habrá terminado ya que la variable tendrá un incremento más. Cuando la variable “i” tenga el valor de 10 y se incremente en uno, la variable “i” tendrá el valor de 11, la cual, al ser evaluada en la condición del ciclo será:

- 11 es menor o igual a 10, como esto es falso, se tomará el camino del “no”, lo que llevará a que se termine el ciclo y también nuestro algoritmo.

Como pudimos observar, nuestro algoritmo reaccionó satisfactoriamente a nuestra prueba de escritorio, lo cual quiere decir que el algoritmo está diseñado de manera adecuada.

Ahora mostremos el código para ver cómo será construido de acuerdo al algoritmo, tenemos que:

- Primero declaramos e inicializamos las variables
- Después solicitamos al usuario ingresar un número
- Lo siguiente a realizar es definir el ciclo for, el cual se divide en 3 partes, la inicialización de la variable, la condición y finalmente el incremento de la variable
- Dentro del ciclo se realiza la multiplicación
- Y también se realiza la impresión de la tabla de multiplicar, donde:
    - Se imprime el número que el usuario ingresó, seguido del signo de "por", este, a su vez por la variable "i", después el signo "igual" y finalmente el resultado, representado por la variable "resultado".

Así quedaría el código para este algoritmo.

Ahora te mostraré la corrida o ejecución del código, como podemos observar, se imprime de manera adecuada la tabla de multiplicar del dos.

Muy bien, este ejemplo es muy bueno para seguir explicando el ciclo for. Haremos una modificación a la variable "i" y cambiaremos su incremento. Ahora, la variable "i" comenzará en 2 e irá incrementando de 2 en 2, es decir, ahora tendremos una tabla del dos pero solo en sus números pares. Para esto, el algoritmo modificado quedaría de la siguiente forma:

- Declaramos las variables que necesitaremos para este problema
- El sistema pedirá al usuario ingresar un número
- El usuario ingresará un número, para este caso, nuevamente 2, 2 representará a la variable "numero"
- Ahora inicializamos la variable "i" en 2

- Después pasamos a nuestra condición; esta no cambia con respecto al algoritmo anterior. Si leemos la condición con nuestro valor de 2 sería de la siguiente forma:
    - 2 es menor o igual a 10, como esto es cierto, continuaremos por el camino del “si”
- Lo siguiente a realizar es llevar a cabo la multiplicación, donde: el resultado es igual a 2 por 2, que es 4
- Posteriormente, se imprimirá el resultado
- Y finalmente se incrementará el valor de la variable “i” en 2, es decir, ahora “i” tiene el valor de 4, y será nuevamente evaluada en nuestro ciclo for.
- Entonces, tenemos que:
    - 4 es menor o igual que 10, como esto es verdadero, continuaremos nuevamente por el camino del “si”
- Lo siguiente es realizar la operación, donde: el resultado es igual a 2 por 4, que es 8
- Posteriormente se imprimirá el resultado
- Y finalmente la variable “i” se incrementará nuevamente en 2; ahora “i” tiene el valor de 6

Nuevamente pudimos observar que en nuestra prueba de escritorio, la variable “i” se estará incrementando, pero ahora en dos; hasta el momento tenemos que la variable “i” vale 6, lo cual nos lleva a concluir que siempre se estará incrementando de dos en dos hasta llegar al 10, lo cual tiene como resultado que la tabla del 2 se imprimirá por completo, pero el ciclo aun no habrá terminado ya que la variable tendrá un incremento más. Cuando la variable “i” tenga el valor de 10 y se incremente en dos, la variable “i” tendrá el valor de 12, la cual, al ser evaluada en la condición del ciclo será:

12 es menor o igual a 10, como esto es falso, se continuará por el camino del “no”, lo que llevará a que se termine el ciclo y también nuestro algoritmo.

Como pudimos observar, nuestro algoritmo reaccionó satisfactoriamente a nuestra prueba de escritorio, lo cual quiere decir que el algoritmo está diseñado de manera adecuada.

Ahora mostremos el código para ver cómo será construido de acuerdo al algoritmo, tenemos que:

- Primero declaramos e inicializamos las variables
- Después solicitamos al usuario ingresar un número
- Lo siguiente a realizar es definir el ciclo for, el cual se divide en 3 partes, la inicialización de la variable, la condición y finalmente el incremento de la variable
- Dentro del ciclo se realiza la multiplicación
- Y también se realiza la impresión de la tabla de multiplicar, donde:
    - Se imprime el número que el usuario ingresó, seguido del signo de “por”, este, a su vez por la variable “i”, después el signo “igual” y finalmente el resultado, representado por la variable “resultado”.

Así quedaría el código para este algoritmo.

Ahora te mostraré la corrida o ejecución del código. Como podemos observar, se imprime de manera adecuada la tabla de multiplicar del dos pero incrementada en dos.

Sigamos con el flujo de este ejemplo y cambiamos nuevamente los valores del ciclo para tener resultados diferentes. Ahora hagamos el incremento en dos pero para números nones, es decir, solo modificaremos la variable “i”, y la inicializamos en 1 nuevamente. Para esto, el algoritmo modificado quedaría de la siguiente forma:

- Declaramos las variables que necesitaremos para este problema
- El sistema pedirá al usuario ingresar un número
- El usuario ingresará un número, para este caso, nuevamente 2. 2 representará a la variable “numero”
- Ahora inicializamos la variable “i” en 1 nuevamente

- Después pasamos a nuestra condición. Esta no cambia con respecto al algoritmo anterior. Si leemos la condición con nuestro valor de 1, sería de la siguiente forma:
    - 1 es menor o igual a 10, como esto es cierto, continuaremos por el camino del “si”
- Lo siguiente a realizar es llevar a cabo la multiplicación, donde: el resultado es igual a 2 por 1, que es 2
- Posteriormente, se imprimirá el resultado
- Y finalmente se incrementará el valor de la variable “i” en 2, es decir, ahora “i” tiene el valor de 3, y será nuevamente evaluada en nuestro ciclo for.
- Entonces, tenemos que:
    - 3 es menor o igual que 10, como esto es verdadero, continuaremos nuevamente por |el camino del “si”
- Lo siguiente es realizar la operación, donde: el resultado es igual a 2 por 3, que es 6
- Posteriormente se imprimirá el resultado
- Y finalmente la variable “i” incrementara nuevamente en 2, ahora “i” vale 5

Nuevamente pudimos observar que en nuestra prueba de escritorio, la variable “i” se estará incrementando de dos en dos. Hasta el momento tenemos que la variable “i” vale 5, lo cual nos lleva a concluir que siempre se estará incrementando de dos en dos hasta llegar al 10, lo cual tiene como resultado que la tabla del 2 se imprimirá por completo, pero el ciclo aun no habrá terminado ya que la variable tendrá un incremento más. Cuando la variable “i” tenga el valor de 9 y se incremente en dos, la variable “i” tendrá el valor de 11, la cual, al ser evaluada en la condición del ciclo será:

11 es menor o igual a 10, como esto es falso, se tomará el camino del “no”, lo que llevará a que se termine el ciclo y también nuestro algoritmo.

Como pudimos observar, nuestro algoritmo reaccionó satisfactoriamente a nuestra prueba de escritorio, lo cual quiere decir que el algoritmo está diseñado de manera adecuada.

Ahora mostremos el código para ver cómo será construido de acuerdo al algoritmo, tenemos que:

- Primero declaramos e inicializamos las variables
- Después solicitamos al usuario ingresar un número
- Lo siguiente a realizar es definir el ciclo for, el cual se divide en 3 partes, la inicialización de la variable, la condición y finalmente el incremento de la variable
- Dentro del ciclo se realiza la multiplicación
- Y también se realiza la impresión de la tabla de multiplicar, donde:
    - Se imprime el número que el usuario ingresó, seguido del signo de "por", este, a su vez por la variable "i" después el signo "igual" y finalmente el resultado, representado por la variable "resultado".

Así quedaría el código para este algoritmo.

Ahora te mostraré la corrida o ejecución del código. Como podemos observar, se imprime de manera adecuada la tabla de multiplicar del dos incrementada en dos, para números nones.

Continuemos con este ejemplo y cambiemos nuevamente los valores del ciclo para tener resultados diferentes. Ahora hagamos un decremento para nuestra tabla del 2, es decir, ahora mostraremos la tabla del dos en sentido contrario, comenzaremos con el 10, después el 9, seguido por el 8 y así hasta llegar al 1. Modificaremos las tres partes del ciclo for para que este pueda funcionar de acuerdo a lo que ahora requerimos. Para esto, el algoritmo modificado quedaría de la siguiente forma:

- Declaramos las variables que necesitaremos para este problema
- El sistema pedirá al usuario ingresar un número
- El usuario ingresará un número, para este caso, nuevamente 2. 2 representará a la variable "numero"
- Ahora inicializamos la variable "i" en 10, esto, para lograr el decremento
- Después pasamos a nuestra condición, esta cambiará con respecto al algoritmo anterior, ahora la variable "i" debe ser mayor o igual a 1. Si leemos la condición con nuestro valor de 10 sería de la siguiente forma:

    - 10 es mayor o igual a 1, como esto es cierto, continuaremos por el camino del “si”
- Lo siguiente a realizar es llevar a cabo la multiplicación, donde el resultado es igual a 2 por 10, que es 20
- Posteriormente, se imprimirá el resultado
- Y finalmente se decrementa el valor de la variable “i” en 1, es decir, ahora “i” vale 9, ya que 10 menos 1 es igual a 9, y será nuevamente evaluada en nuestro ciclo for.
- Entonces, tenemos que:
    - 9 es mayor o igual que 1, como esto es verdadero, tomaremos nuevamente el camino del “si”
- Lo siguiente es realizar la operación, donde: el resultado es igual a 2 por 9, que es 18
- Posteriormente se imprimirá el resultado
- Y finalmente la variable “i” decrementa nuevamente en 1, ahora “i” vale 8

Nuevamente pudimos observar que en nuestra prueba de escritorio, la variable “i” se estará decrementando de uno en uno. Hasta el momento tenemos que la variable “i” vale 8, lo cual nos lleva a concluir que siempre se estará decrementando de uno en uno hasta llegar al 1, lo cual tiene como resultado que la tabla del 2 se imprimirá por completo, ahora de manera invertida, pero, el ciclo aun no habrá terminado ya que la variable tendrá un decremento más. Cuando la variable “i” tenga el valor de 1 y se decremente en uno, la variable “i” tendrá el valor de 0, la cual, al ser evaluada en la condición del ciclo será:

0 es mayor o igual a 1, como esto es falso, continuará por el camino del “no”, lo que llevará a que se termine el ciclo y también nuestro algoritmo.

Como pudimos observar, nuestro algoritmo reaccionó satisfactoriamente a nuestra prueba de escritorio, lo cual quiere decir que el algoritmo está diseñado de manera adecuada.

Ahora mostremos el código para ver cómo será construido de acuerdo al algoritmo, tenemos que:

- Primero declaramos e inicializamos las variables
- Después solicitamos al usuario ingresar un número
- Lo siguiente a realizar es definir el ciclo for, el cual se divide en 3 partes, la inicialización de la variable, la condición y finalmente el decremento de la variable
- Dentro del ciclo se realiza la multiplicación
- Y también se realiza la impresión de la tabla de multiplicar, donde:
    - Se imprime el número que el usuario ingresó, seguido del signo de "por", este, a su vez por la variable "i", después el signo "igual" y finalmente el resultado, representado por la variable "resultado".

Así quedaría el código para este algoritmo.

Ahora te mostraré la corrida o ejecución del código. Como podemos observar, se imprime de manera adecuada la tabla de multiplicar del dos, la cual decrementó en uno.

Sigamos con este ejemplo y cambiamos nuevamente los valores del ciclo para tener resultados diferentes. Ahora hagamos un decremento para nuestra tabla del 2, pero para sus números pares, es decir, ahora mostraremos la tabla del dos en sentido contrario, comenzando con el 10, después el 8, seguido por el 6 y así hasta llegar al 2. Del algoritmo anterior únicamente modificaremos la parte del ciclo for donde se realiza el decremento. Para esto, el algoritmo modificado quedaría de la siguiente forma:

- Declaramos las variables que necesitaremos para este problema
- El sistema pedirá al usuario ingresar un número
- El usuario ingresará un número, para este caso, nuevamente 2. 2 representará a la variable "numero"
- Ahora inicializamos la variable "i" en 10, esto, para lograr el decremento
- Después pasamos a nuestra condición, esta, no cambiará con respecto al algoritmo anterior, tenemos nuevamente que la variable "i" debe ser mayor o igual a 1 si leemos la condición con nuestro valor de 10 sería de la siguiente forma:

    - 10 es mayor o igual a 1, como esto es cierto, continuaremos por el camino del “si”
- Lo siguiente a realizar es llevar a cabo la multiplicación, donde: el resultado es igual a 2 por 10, que es 20
- Posteriormente, se imprimirá el resultado
- Y finalmente se decrementa el valor de la variable “i” en 2, es decir, ahora “i” vale 8, ya que 10 menos 2 es igual a 8, y será nuevamente evaluada en nuestro ciclo for.
- Entonces, tenemos que:
    - 8 es mayor o igual que 1, como esto es verdadero, tomaremos nuevamente el camino del “si”
- Lo siguiente es realizar la operación, donde: el resultado es igual a 2 por 8, que es 16
- Posteriormente se imprimirá el resultado
- Y finalmente la variable “i” decrementa nuevamente en 2, ahora “i” tiene el valor de 6

Nuevamente pudimos observar que en nuestra prueba de escritorio, la variable “i” se estará decrementando de dos en dos. Hasta el momento tenemos que la variable “i” tiene el valor de 6, lo cual nos lleva a concluir que siempre se estará decrementando de dos en dos hasta llegar al 2, lo cual tiene como resultado que la tabla del 2 se imprimirá por completo, ahora de manera invertida, pero, el ciclo aún no habrá terminado, ya que la variable tendrá un decremento más. Cuando la variable “i” tenga el valor de 2 y se decremente en dos, la variable “i” tendrá el valor de 0, la cual, al ser evaluada en la condición del ciclo será:

0 es mayor o igual a 1, como esto es falso, se tomará el camino del “no”, lo que llevará a que se termine el ciclo y también nuestro algoritmo.

Como pudimos observar, nuestro algoritmo reaccionó satisfactoriamente a nuestra prueba de escritorio, lo cual quiere decir que el algoritmo está diseñado de manera adecuada.

Ahora mostremos el código para ver cómo será construido de acuerdo al algoritmo, tenemos que:

- Primero declaramos e inicializamos las variables
- Después solicitamos al usuario ingresar un número
- Lo siguiente a realizar es definir el ciclo for, el cual se divide en 3 partes, la inicialización de la variable, la condición y finalmente el decremento de la variable
- Dentro del ciclo se realiza la multiplicación
- Y también se realiza la impresión de la tabla de multiplicar, donde:
    - Se imprime el número que el usuario ingresó, seguido del signo de "por", este, a su vez por la variable "i", después el signo "igual" y finalmente el resultado, representado por la variable "resultado".

Así quedaría el código para este algoritmo.

Ahora te mostraré la corrida o ejecución del código. Como podemos observar, se imprime de manera adecuada la tabla de multiplicar del dos, la cual decrementó en dos.

Bueno, hasta aquí este video dedicado a explicar el ciclo for. Espero te haya sido útil y que hayas comprendido el concepto y funcionamiento de esta estructura de control. Hasta el siguiente video.

3.8 Video 5. Do-While

Para esta estructura de control, se retomó el ejemplo del if simple, pero no se enfatizó en el ejemplo como tal, sino, que se enfatiza en el funcionamiento de esta nueva estructura de control. Para este ejemplo se anexó una actividad más, la cual da la opción de ingresar otra calificación o no, donde, de ser verdadera la respuesta del usuario, se repetiría nuevamente todo el procedimiento, propiciando así un ciclo, hasta que el usuario ingresara un "no" como respuesta.

Se explica cómo se crea el código basándose en el algoritmo y finalmente se muestra la ejecución del mismo.

Para este ejemplo se altera el diagrama, sufriendo una pequeña modificación en la condición del ciclo, donde ahora se tendrá que ingresar un “1” para representar un “si” o un “2” para representar un “no”. Posteriormente se explicará la creación del código basado en este algoritmo nuevo, y finalmente la ejecución del código.

Cabe mencionar que en el video se explica la diferencia entre el ciclo “Do-While” y el ciclo “While” de manera muy visual.

Para concluir, se tiene el script de esta estructura de control, en la cual se describen de forma puntual los elementos que conforman este diagrama, así como también su funcionamiento.

3.8.1 Do-While simple

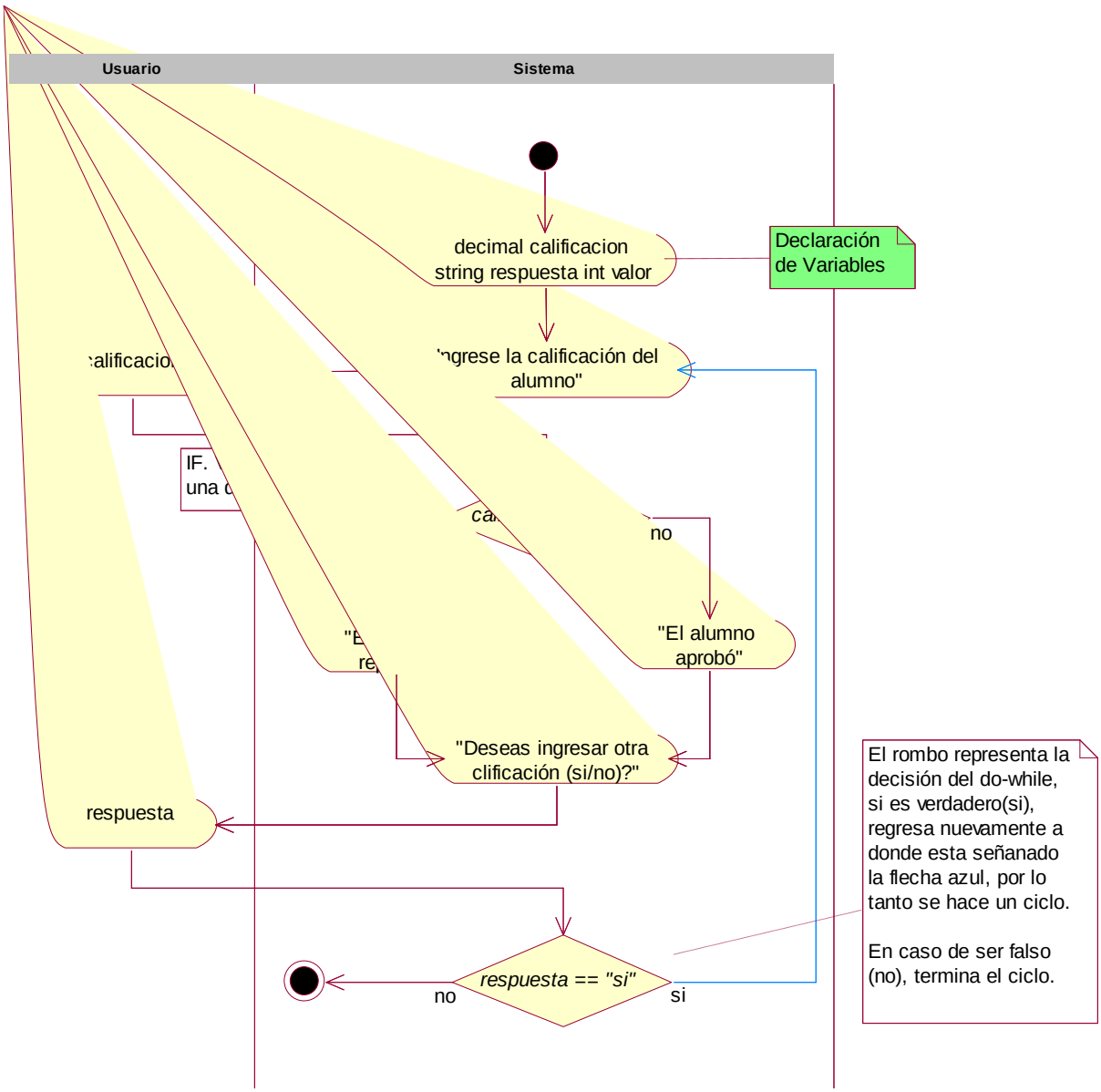

Figura 19. Diagrama de actividades Do-While simple

3.8.2 Código en C#

```
static void Main(string[] args)
        {
            //Declaración de Variables
            string respuesta;
            double calificacion;

            //Comienza el ciclo do-While
            do
            {
                //Solicita la calificación al usuario
                Console.WriteLine("Ingrese la calificación del alumno:");
                respuesta = Console.ReadLine();
                calificacion = Convert.ToDouble(respuesta);

                //Verificamos si es una calificación aprobatoria o no
                //Donde: Aprobatoria es de 7 a 10 y reprobatoria de 0 a 6.99
                if (calificacion < 7)
                {
                    //Mostramos el resultado al usuario
                    Console.WriteLine("El alumno reprobó");
                }
                else
                {
                    //Mostramos el resultado al usuario
                    Console.WriteLine("El alumno aprobó");
                }

                //Solicita la opción al usuario
                Console.WriteLine("Deseas ingresar otra calificación (si/no)?:");
                respuesta = Console.ReadLine();

            } while (respuesta == "si");
        }
```

3.8.3 Resultado de la ejecución del código en C#

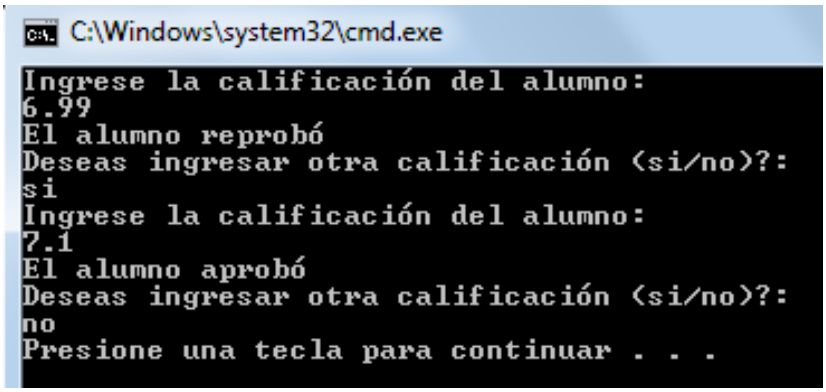

Figura 20. Resultado de la ejecución del código Do-While simple

3.8.4 Do-While

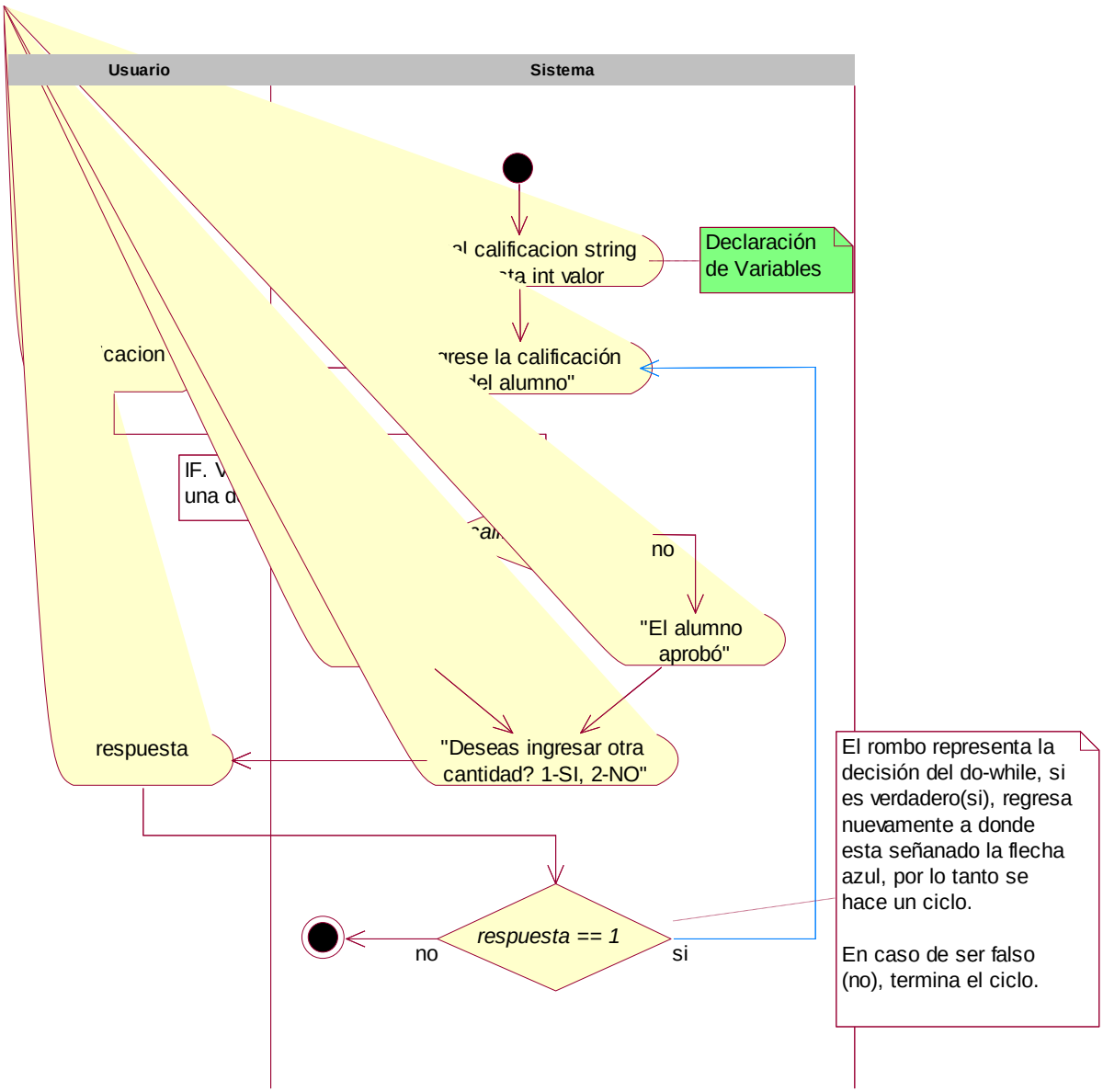

Figura 21. Diagrama de actividades Do-While

3.8.5 Código en C#

```
static void Main(string[] args)
        {
            //Declaración de Variables
            string respuesta;
            double calificacion;
            int valor;

            //Comienza el ciclo do-While
            do
            {
                //Solicita la calificación al usuario
                Console.WriteLine("Ingrese la calificación del alumno:");
                respuesta = Console.ReadLine();
                calificacion = Convert.ToDouble(respuesta);

                //Verificamos si es una calificación aprobatoria o no
                //Donde: Aprobatoria es de 7 a 10 y reprobatoria de 0 a 6.99
                if (calificacion < 7)
                {
                    //Mostramos el resultado al usuario
                    Console.WriteLine("El alumno reprobó");
                }
                else
                {
                    //Mostramos el resultado al usuario
                    Console.WriteLine("El alumno aprobó");
                }

                //Solicita la opción al usuario
                Console.WriteLine("Deseas ingresar otra calificación? 1-SI, 2-NO:");
                respuesta = Console.ReadLine();
                valor = Convert.ToInt32(respuesta);

            } while (valor == 1);
}
```

3.8.6 Resultado de la ejecución del código en C#

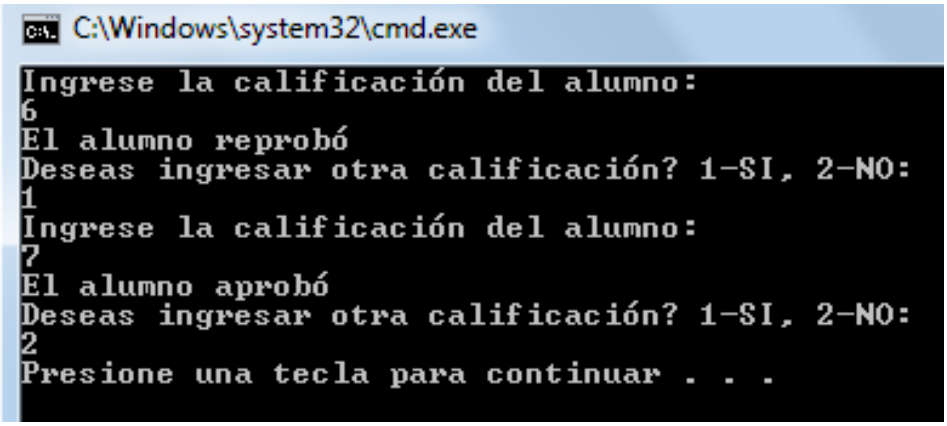

Figura 22. Resultado de la ejecución del código Do-While

3.8.7 Script

Hola mi nombre es Román, estudié la carrera de Ing. de Software en la Universidad Madero. Nuevamente estoy contigo para esta serie de videos.

Es importante recordarte que si tienes dudas respecto a algún término, regreses y visualices el primer video y el segundo video en donde se explican a detalle los términos y conceptos de lo visto durante estos videos, además de explicarte los componentes y significados de las diferentes figuras que se emplean en los diagramas.

De estar todo claro continuemos con este video. Para este video abordaremos a la estructura de control Do-While. Esta estructura funciona como un ciclo, es decir, podremos repetir una serie de instrucciones si así se desea. Una característica importante de esta estructura de control es que el código que contenga esta estructura se ejecutará al menos una vez, y dependiendo de la condición del do-while, si es verdadera, se ejecutará nuevamente el ciclo, o en caso de ser falsa, termina el ciclo. Debemos aclarar que también existe otra estructura de control llamada While, ésta también es un ciclo, y funciona de manera similar al Do-While.

Nota: Agregar los dos diagramas en el video para la comparación

Te mostraré dos diagramas para explicarte la diferencia entre ellos. El Do-While en su diagrama está representado por un rombo, donde la flecha verdadera regresa a alguna actividad arriba de él, provocando que nuevamente se realicen una serie de instrucciones que ya se realizaron previamente, por esto se realiza un ciclo, a diferencia del While, donde es el caso contrario, la flecha verdadera baja y ejecuta el código que contiene y no regresa como el Do-While.

Bien, regresando al uso del ciclo del Do-While, te mostraré un ejemplo explicándote su funcionamiento. Para esto, retomaremos el ejemplo del if, en donde se solicita al usuario una calificación y se indicaba si el alumno aprobó o reprobó.

El problema sería el siguiente: Tomando el ejemplo del if, modificar el algoritmo, para que este, muestre la opción de “Ingresar otra calificación”.

Como en los ejemplos anteriores, lo primero que se realiza es:

- Colocar un swimlane para lo que realizará el usuario y otro para lo que realizará el sistema
- En el swimlane del sistema comenzamos el diagrama colocando un círculo negro, el cual indica el inicio del diagrama.
- Lo siguiente a realizar es la declaración de las variables
- Seguido a esto, el usuario ingresará la calificación
- El valor de la variable “calificacion” pasará a ser evaluada en nuestro if, donde, de ser verdadera imprimirá al usuario “El alumno reprobó”, y de ser falsa imprimirá al usuario “El alumno aprobó”.
- Ahora, el sistema pedirá al usuario si desea ingresar otra calificación
- El usuario ingresará una respuesta, ya sea un “si” o un “no”, es aquí donde inicia el ciclo do-while, ya que, de ser la respuesta un “si”, se regresará hasta la parte del diagrama donde se pide al usuario ingresar una calificación. Esto se muestra claramente con la flecha de color azul. El proceso continuará de

manera normal, hasta llegar nuevamente a la condición del do-while, donde se evaluará el tipo de respuesta. Si se responde nuevamente con un “si”, el proceso se repetirá nuevamente haciendo un ciclo, pero si se responde con un “no” el algoritmo terminará, es decir, termina el ciclo.

Ahora mostraré el código para que puedas visualizar cómo quedaría de acuerdo al diagrama:

- Primero declaramos las variables
- Ahora, comienza la declaración de la sintaxis del ciclo do-while, donde se escribe el “do”
- Después pedimos al usuario ingresar una calificación
- Verificamos en nuestro if si la calificación es aprobatoria o no,
- Seguido a esto, se imprime al usuario una respuesta
- Después preguntamos al usuario si desea ingresar otra calificación
- Finalmente se evalúa la condición de acuerdo a la respuesta en el do-while, donde, de ser un “si”, se estará repitiendo convirtiéndose en un ciclo.

Ahora te mostraré la ejecución o corrida del código. Como puedes observar, después de haber ingresado una calificación, el programa nos muestra la opción de ingresar otra calificación. Al ingresar un “si”, se estará repitiendo lo mismo, y esto terminará cuando se ingrese como respuesta un “no”.

Como se puede ver, el ciclo do-while es bastante sencillo. Realizaré una pequeña modificación al diagrama en la parte del ciclo do-while, y esta quedará de la siguiente manera:

- Iniciamos de manera normal, colocando un swimlane para lo que realizará el usuario y otro para lo que realizará el sistema
- En el swimlane del sistema comenzamos el diagrama poniendo un círculo negro, el cual indica el inicio del diagrama.
- Lo siguiente a realizar es la declaración de las variables
- Seguido a esto, el usuario ingresará la calificación

- Esta calificación pasará a ser evaluada en nuestro if, donde, de ser verdadera imprimirá al usuario “El alumno reprobó”, y de ser falsa imprimirá al usuario “El alumno aprobó”.
- Ahora, el sistema pedirá al usuario si desea ingresar otra calificación
- El usuario ingresará una respuesta; aquí es donde se realizará la modificación que se mencionó con anterioridad. En lugar de esperar como respuesta un “si” o un “no”, ahora determinaremos que el usuario ingrese un “1” para referirse a un “si” o un ”2” para referirse a un “no”. De ser la respuesta un “1”, se regresará hasta la parte del diagrama donde se pide al usuario ingresar una calificación. Esto se muestra claramente con la flecha de color azul. El proceso continuará de manera normal, hasta llegar nuevamente a la condición del do-while, donde se evaluará la respuesta. Si se responde nuevamente con un “1”, el proceso se repetirá nuevamente haciendo un ciclo, pero si se responde con un “2”, es decir un “no”, el algoritmo terminará, es decir, termina el ciclo.

Ahora mostraré el código para que puedas visualizar cómo quedaría de acuerdo al diagrama:

- Primero declaramos las variables
- Ahora, comienza la declaración de la sintaxis del ciclo do-while, donde se escribe el “do”
- Después pedimos al usuario ingresar una calificación
- Verificamos en nuestro if si la calificación es aprobatoria o no,
- Seguido a esto, se imprime al usuario una respuesta
- Después preguntamos al usuario si desea ingresar otra calificación
- Finalmente se evalúa la respuesta en el do-while, donde, de ser un “1”, se estará repitiendo convirtiéndose en un ciclo.

Ahora te mostraré la ejecución o corrida del código. Como puedes observar, después de haber ingresado una calificación, el programa nos muestra la opción de ingresar otra calificación. Al ingresar un “1”, se estará repitiendo lo mismo; esto terminará cuando se ingrese como respuesta un “2”.

Muy bien, hasta aquí este video dedicado a explicar el ciclo Do-While. Espero te haya sido útil y que hayas comprendido el concepto y funcionamiento de esta estructura de control. Hasta el siguiente video.

3.9 Video 6. While

Finalmente se tiene una última estructura de control en la que como todas las demás, se planteó un ejemplo. En esta, dependiendo de la opción que se ingrese, se realizará cierta acción, es decir, para cada respuesta se tiene un proceso a ejecutar. Si se ingresa la opción de “1”, se calculará el área de un cuadrado; de elegirse la opción “2”, se calculará el perímetro del cuadrado y de ingresarse el valor de “3”, el algoritmo culminará.

Para su prueba de escritorio, se evalúan cada una de las opciones correspondientes, con la finalidad de verificar si el algoritmo se realizó de manera correcta.

También se muestra el correcto flujo de creación del código mediante el algoritmo diseñado, y una vez que se tenga el código, se ejecutará para ver los posibles resultados.

Todo este procedimiento es bien explicado en el script correspondiente a este video final, donde se describe paso a paso cómo debe crearse el diagrama de actividades, cómo se debe probar el algoritmo con las pruebas de escritorio, cómo elaborar el código para este problema basado en el diagrama que se diseñó y finalmente la ejecución del código.

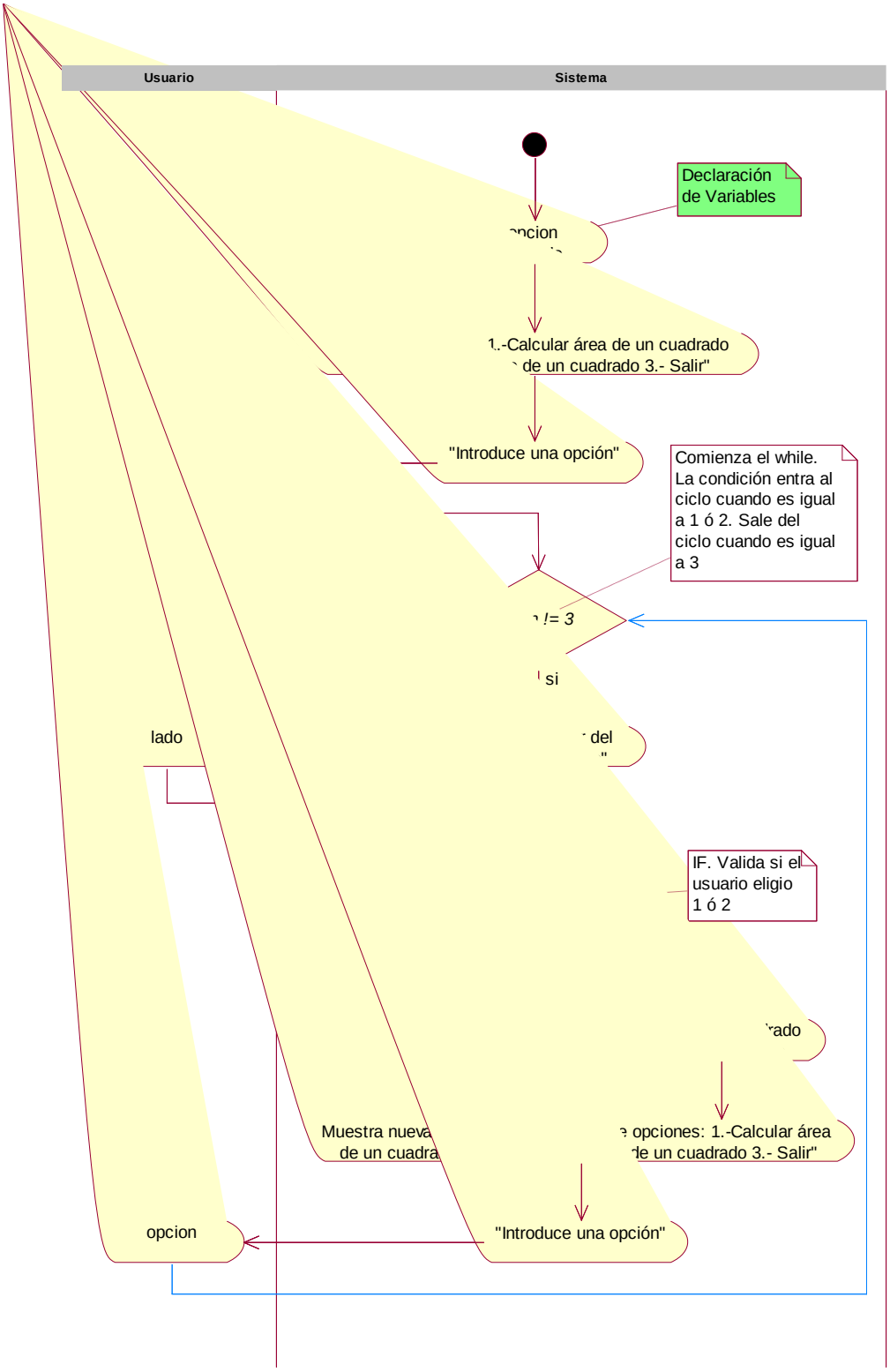

Figura 23. Diagrama de actividades While

3.9.1 Prueba de escritorio

Prueba de Escritorio		
# Prueba	Variable	Resultado
1	1	Imprime Área
2	2	Imprime Perímetro
3	3	Termina el ciclo

Tabla 18. Prueba de escritorio While

3.9.2 Código en C#

```
static void Main(string[] args)
        {
            //Declaracion de Variables
            int opcion;
            int lado;

            //Se muestra el memú de opciones
            Console.WriteLine("Menú de opciones:");
            Console.WriteLine("1.- Calcular el área de un cuadrado");
            Console.WriteLine("2.- Calcular el perímetro de un cuadrado");
            Console.WriteLine("3.- salir");
            Console.WriteLine("Introduce una opción:");
            opcion = int.Parse(Console.ReadLine());

            //Comienza el While, dónde, si la opcion es difernete de 3 entra al
ciclo
            //Si la opción es igual a 3, termina el ciclo
            while (opcion != 3)
            {
                //Solicitamos ingresar el valor del lado del cuadrado
                Console.WriteLine("Introduce el valor del lado del cuadrado");
                lado = int.Parse(Console.ReadLine());

                //Esto es un if, lo empleamos para verificar que número ingreso
                //Si eligío el número uno, calcula el área
                //Si eligío el número dos, calcula el perímetro
                if (opcion == 1)
                {
                    //Se muestra el resultado al usuario
                    Console.WriteLine("El área del cuadrado es: {0}", lado * lado);
                }else
                {
                    //Se muestra el resultado al usuario
```

```
                    Console.WriteLine("El perímetro del cuadrado es: {0}", lado *
4);
                }

                //Se muestra el memú de opciones
                Console.WriteLine("Menú de opciones:");
                Console.WriteLine("1.- Calcular el área de un cuadrado");
                Console.WriteLine("2.- Calcular el perímetro de un cuadrado");
                Console.WriteLine("3.- salir");
                Console.WriteLine("Introduce una opción:");
                opcion = int.Parse(Console.ReadLine());
            }
        }
```

3.9.3 Resultado de la ejecución del código en C#

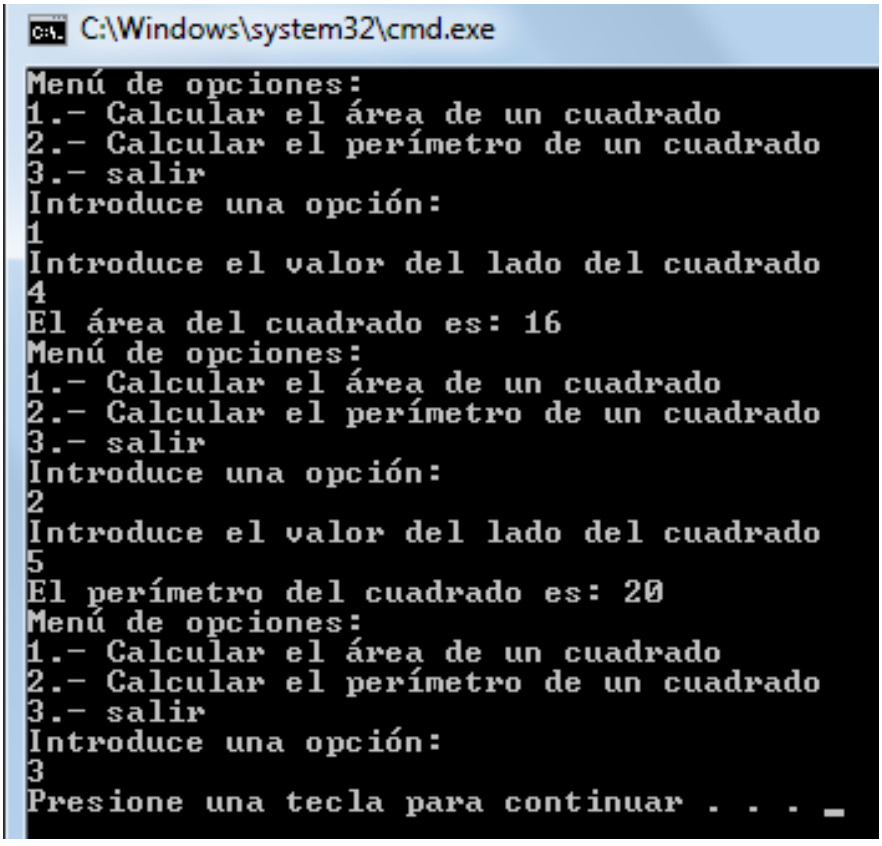

Figura 24. Resultado de la ejecución del código While

3.9.4 Script

Hola mi nombre es Román. Estudié la carrera de Ingeniería de Software en la Universidad Madero. Nuevamente estoy contigo para esta serie de videos.

Es importante recordarte que si tienes dudas respecto a algún término, regreses y visualices el primer video y el segundo video en donde se explican a detalle los términos y conceptos de lo visto durante estos videos, además de explicarte los componentes y significados de las diferentes figuras que se emplean en los diagramas.

Antes de continuar con el presente video del While, te comento que en el video correspondiente al Do-While se explica la diferencia estos dos tipos de ciclos, por lo que te recomiendo ver antes de este video, el del Do-While.

De estar todo claro continuemos con este video. Para este video abordaremos a la estructura de control While. Esta estructura está representada por un rombo y funciona como un ciclo, es decir, podremos repetir una serie de instrucciones si la condición es verdadera y sale del ciclo al ser falsa. Bien, ahora te mostraré un ejemplo explicándote el funcionamiento del ciclo While.

Problema: Elaborar un algoritmo a manera de diagrama de actividades en el cual a manera de menú pida la opción de calcular el área o perímetro de un cuadrado, además de tener la opción de salir.

Bien lo primero que realizaremos será:

- Colocar un swimlane para lo que realizará el usuario y otro para lo que realizará el sistema
- En el swimlane del sistema comenzamos el diagrama colocando un círculo negro, el cual indica el inicio del diagrama.
- Lo siguiente a realizar es la declaración de las variables
- Ahora se mostrará el menú con las opciones
- Después se pedirá al usuario ingresar una opción
- El usuario ingresa la opción

- Seguido a esto, la opción pasará a ser evaluada por la condición del While, donde, si la opción es diferente de 3 continuará por el camino del “si”, es decir, habrá elegido el 1 (calcular el área) o el 2 (calcular el perímetro), pero si la opción es igual al 3, nuestro algoritmo terminará. Cabe mencionar que en el algoritmo, se está dando por hecho que sólo se ingresarán valores como el 1, 2 y 3, ya que este algoritmo no está diseñado para cuando se ingresen otros tipos de valores, ya que, para este caso, no es el objetivo demostrarlo.
- Ahora, si se toma el camino del “si”, el algoritmo pedirá al usuario que ingrese un valor para el lado del cuadrado, esto, dependiendo del paso anterior.
- El usuario ingresará un valor para el lado del cuadrado.
- Bien, previamente, cuando el usuario ingresó una opción de lo que quería realizar, en la variable “opcion” se guardó un valor, ya sea el 1 ó 2, porque de haber sido el 3 ya habría terminado el algoritmo. Entonces tanto el valor de “1” como “2” pasarán a ser evaluados en el siguiente rombo. Este rombo representa a un if, el rombo anterior representaba al ciclo While. En este if se evaluará la siguiente condición: opción es igual a 1, donde de ser verdadera se continuará por el camino del “si” y se calculará el área del cuadrado, pero de ser falsa, se tomará el camino del “no” y se calculará el perímetro del cuadrado.
- Finalmente, independientemente de la opción que se tome, se seguirá con el flujo del diagrama y se mostrará nuevamente el menú de opciones
- Se pedirá al usuario ingresar una opción nuevamente, provocando que el ciclo se realice. Este se muestra claramente con la flecha de color azul, la cual regresa hasta el rombo del while.

Bien ahora realicemos una prueba de escritorio para cada uno de los posibles casos. Comencemos para cuando el usuario ingrese un “1”:

- Primero, el sistema muestra el menú de opciones
- Seguido a esto, el sistema pedirá ingresar una opción
- Lo siguiente le corresponde al usuario, el cual ingresará la opción, que para esta primera prueba será “1”. Ahora “1” representa a la variable opción.

- Una vez que se tiene el valor de la variable, ésta pasará a ser evaluada en el While, donde:
    - 1 es diferente de 3; como esto es verdadero, se tomará el camino del “si”
- Lo cual nos lleva a que el sistema nos pida ahora un número, el cual representará a la variable “lado. Ejemplifiquemos que para esta variable el usuario ingresó un 4; ahora 4, representa a la variable “lado”
- Una vez ingresado este valor, continuamos con el flujo del algoritmo, lo cual nos lleva a evaluar a la variable “opcion” en el if. Recordemos, que la variable opción tiene el valor de “1”, para lo cual tendríamos que:
    - 1 es igual a 1, como esto es cierto, entonces continuaremos por el camino del “si”, para lo cual el sistema calculará el valor del área del cuadrado e imprimirá ese valor al usuario. El cálculo se realiza dentro de la impresión, para lo cual tenemos que 4 por 4 es igual a 16
- Después de haber impreso el resultado, el algoritmo mostrará nuevamente el menú, y pedirá ingresar nuevamente una opción.
- El usuario ingresará una opción. Esta opción será nuevamente evaluada en el While, convirtiéndose en un ciclo. Como se muestra con la flecha de color azul.

Ahora, aprovechemos este ciclo, para realizar nuestra segunda prueba de escritorio, para lo cual teníamos que el usuario ingresaría una opción. Supongamos que ahora el usuario ingresará el número 2, es decir, elegirá en el menú la opción 2, entonces:

- Una vez que se tiene la variable, esta pasará a ser evaluada en el while, donde:
    - 2 es diferente de 3, como esto es cierto, se tomará el camino del “si”
- Lo cual nos lleva a que el sistema nos pida ahora un número, el cual representará a la variable “lado”. Ejemplifiquemos que para esta variable el usuario ingresó un 5. Ahora 5, representa a la variable “lado”
- Una vez ingresado este valor, continuamos con el flujo del algoritmo, lo cual nos lleva a evaluar a la variable “opción” en el if. Recordemos, que la variable opción tiene el valor de “2”, para lo cual tendríamos que:

    - 2 es igual a 1, como esto no es cierto, entonces continuaremos por el camino del “no”, para lo cual el sistema calculará el valor del perímetro del cuadrado e imprimirá ese valor al usuario. El cálculo se realiza dentro de la impresión, para lo cual tenemos que 5 por 4 es igual a 20
- Después de haber impreso el resultado, el algoritmo mostrará nuevamente el menú, y pedirá ingresar nuevamente una opción.
- El usuario ingresará una opción. Esta opción será nuevamente evaluada en el While, convirtiéndose en un ciclo. Como se muestra con la flecha de color azul.

Nuevamente aprovechemos este ciclo, para realizar nuestra tercera prueba de escritorio, para lo cual teníamos que el usuario ingresaría una opción, supongamos que ahora el usuario ingresará el número 3, es decir, elegirá en el menú la opción 3, entonces:

- Una vez que se tiene la variable, esta pasará a ser evaluada en el while, donde:
    - 3 es diferente de 3, como esto no es cierto, se continuará por el camino del “no” y terminará nuestro algoritmo.

Ahora te mostraré cómo quedaría el código en C# para este algoritmo, que, de acuerdo a este diagrama se construiría de la siguiente manera:

- Se tiene la declaración de las variables
- Seguido a esto se muestra el menú de opciones
- Después, comienza el While, donde de cumplirse la condición se ejecutaría el código en su interior
- El paso siguiente es solicitar al usuario que ingrese un valor correspondiente al lado del cuadrado
- Después el valor de la variable opción pasará a ser evaluado en el if, donde de haberse elegido el “1” se calculará e imprimirá el área del cuadrado, o de haberse elegido el “2” se calculará e imprimirá el perímetro del cuadrado
- Finalmente, se muestra nuevamente el menú de opciones al usuario.

Ahora te mostraré la ejecución del código respecto a las pruebas de escritorio. Como puedes observar, esta ejecución es exactamente igual a los resultados de las pruebas de escritorio, para lo cual podemos decir que nuestro algoritmo, está bien diseñado.

Como se puede observar esta estructura de control es bastante sencilla, y fácil de aplicar.

Bien, hasta aquí este video, espero que haya sido de ayuda para ti, y que esta estructura haya quedado entendida. Hasta la próxima.

3.10 Pruebas de efectividad del material multimedia

Al término de la edición de los materiales para la primera estructura de control, correspondiente al If, fue necesario corroborar que el material desarrollado cumpliera con el objetivo para el cual fue elaborado: servir como apoyo para los estudiantes de nuevo ingreso. Fue por este motivo, que el material desarrollado fue probado con estudiantes de la Universidad Madero, específicamente de la carrera de Diseño Gráfico, es decir, fue contrastado con alumnos cuyo perfil es completamente diferente a los de las carreras de Ingeniería de Software e Ingeniería en Tecnologías de Información e Internet.

El resultado fue satisfactorio, ya que los estudiantes que probaron este material comprendieron los conceptos incluidos en esta primera estructura, elementos del algoritmo y diagrama de actividades, las pruebas de escritorio y cómo fue que se tradujo el algoritmo a código en C#. Lo anterior permitió inferir que si un alumno que no tiene el perfil de las carreras entienden los conceptos y elementos contenidos, con mucho más razón los estudiantes que sí cuentan con este.

Capítulo 4. Conclusiones, Recomendaciones y Trabajos Futuros

4.1 Métricas

Capítulo		Real		Costo por hora	Costo del tutorial
		Min.	Hrs.		
Capítulo 1		790	13.16666667	$ 120.00	$ 1,580.00
Capítulo 2		1457	24.28333333	$ 120.00	$ 2,914.00
Capítulo 3	Video 1	1268	21.13333333	$ 120.00	$ 2,536.00
	Video 2	1831	30.51666667	$ 120.00	$ 3,662.00
	Video 3	695	11.58333333	$ 120.00	$ 1,390.00
	Video 4	942	15.7	$ 120.00	$ 1,884.00
	Video 5	443	7.383333333	$ 120.00	$ 886.00
	Video 6	637	10.61666667	$ 120.00	$ 1,274.00
	Página Web	1271	21.18333333	$ 120.00	$ 2,542.00
	Doc Capítulo 3	10.14	0.169	$ 120.00	$ 20.28
Capítulo 4		915	15.25	$ 120.00	$ 1,830.00
	TOTAL:			**TOTAL:**	**$ 20,518.28**
Overhead		1270	21.16666667	$ 120.00	$ 2,540.00
		Total General:			**$ 23,058.28**

Figura 25. Métricas de tiempo y costo

4.2 Conclusiones

Para este proyecto de investigación se tienen varias conclusiones, las cuales se listan a continuación:

- El material multimedia desarrollado funciona adecuadamente para los alumnos que tengan acceso a éste, ya que al término de desarrollar el script del if fue probado con alumnos de la universidad que estudiaban otra carrera. Los resultados fueron satisfactorios, ya que estos que cuentan con un perfil diferente, pudieron comprender los conceptos explicados, además de los elementos que este primer script involucraba.
- Para garantizar que los videos del material multimedia realmente puedan apoyar al aprendizaje de los alumnos, se deben elaborar siguiendo de forma correcta el proceso para la resolución de problemas, es decir, primero se muestra el problema al que tendrán que darle solución, después tendrá que realizar el algoritmo en forma de diagrama de actividades. Una vez realizado el diagrama, tendrá que comprobar su funcionamiento mediante pruebas de escritorio para diferentes escenarios al que se enfrentará el algoritmo. Finalmente se muestra la forma en que el algoritmo se convierte en código C#.
- El sustentante aún no cuenta con la experiencia necesaria para calcular los tiempos estimados en la realización de diferentes actividades, ya que se observó un gran desfase de tiempos respecto a lo que se planeó en un principio con lo que en realizad se llevó a cabo.

4.3 Recomendaciones para aprovechar el tutorial

Dentro de algunas recomendaciones para el buen aprovechamiento del material multimedia desarrollado se tienen las siguientes:

- Pausar los videos las veces que sean requeridas para un correcto entendimiento del tema.

- Realizar anotaciones si lo creyere necesario.
- Descargar los materiales disponibles y utilizarlos durante la reproducción de los videos.
- Realizar los ejercicios que se muestran en los videos.

4.4 Trabajos futuros

Como parte de las actividades y trabajos que pueden realizarse a futuro como seguimiento de este proyecto, se encuentran los siguientes:

- Realizar una versión extendida de este material, es decir, realizar un script a detalle para las demás estructuras de control, así como como se hizo en el script del if.
- Continuando con el punto anterior, se pueden incluir ejercicios más complejos que los que se realizaron en los diferentes videos.
- Sería recomendable complementar el presente tutorial, ahora con las estructuras de control interactuando entre sí, por ejemplo un while dentro de una sentencia if o un if dentro de la estructura for.
- Se pueden realizar videos interactivos con los estudiantes, una modalidad que se ocupa en YouTube, en los cuales se plantean escenarios y dependiendo de la respuesta se proyecta una fracción de video.
- En las carreras de Ingeniería Industrial e Ingeniería Mecatrónica se tiene actualmente un curso similar donde los alumnos aprenden algoritmos, uso de diagramas de actividades, Rational Rose y programación, además de otros temas. Como parte de los trabajos futuros, se podría desarrollar un tutorial específicamente para dicha materia, mostrando el uso del lenguaje de programación Pseint. Debido a que este lenguaje es tan sencillo, se podría utilizar a nivel secundaria y preparatoria, por lo que este tutorial también serviría en dichos niveles educativos.

4.5 Lecciones aprendidas

Como parte de las lecciones aprendidas durante la elaboración de la presente investigación, se encuentran las siguientes:

- Para la realización de los videos, lo correcto es grabar el audio en formato mp3, ya que eso facilitará que cuando se realice el render de los videos, sea mucho más rápido el proceso. Además, como únicamente es la voz lo que está almacenado en el audio, no es necesario un formato con mayor calidad, y por consecuencia, mayor tamaño (peso).
- Para complementar el punto anterior, se generaron las primeras versiones de audio en formato WAV, lo cual fue incorrecto, ya que dicho formato es más usado para audio musical.
- Cuando se realiza la edición de los videos en Camtasia, es importante revisar que no se haya movido algún elemento (flechas, imágenes, marcadores) de los utilizados en el video, ya que al finalizar el render y visualizar el video, será necesario hacerlo nuevamente, y por lo tanto, se tendrá que retrabajar.
- Es importante tomar en cuenta los tiempos que se tienen marcados como límite y que se establecieron en el diagrama de Gantt, y con base en ellos, hacer una correcta planeación de las diferentes actividades a realizar, ya que durante el desarrollo de la tesis no se tomó en cuenta que el sustentante no era el único al que le revisaban los materiales y capítulos realizados. Esto provocó que los tiempos se vieran afectados, ya que se requería del segundo capítulo para poder realizar el primer video del material.
- Se debe destinar un tiempo adecuado para la investigación de la información, ya que, como no se planeó cuidadosamente, retrasó los tiempos una semana para la investigación de “Algoritmos”, cuando se había planeado para dos días. Esto servirá para asentar de manera real los tiempos planeados.
- Al principio se tenía una idea muy vaga de lo que implicaba realizar esta tesis, pues el sustentante suponía que solo consistía en elaborar unos videos, los materiales de apoyo y subirlo a la plantilla. Esto no resultó ser así, ya que al realizar el script del if, se apreció la profundidad de este proyecto. Se tenía

que provocar que el alumno de nuevo ingreso aprendiera de forma muy simple y sencilla, pero a la vez sin hacer tedioso el aprendizaje. Ayudar al alumno a desarrollar su lógica computacional mediante un material visual, este a su vez multimedia y que tuviera una explicación detallada de los algoritmos básicos que se empleaban, lo cual incrementaba la complejidad del proyecto.

- Se deben utilizar fuentes de información 100% confiables y en especial Ebsco. Se observó que tiene una funcionalidad muy extensa y que se puede encontrar prácticamente cualquier tema allí. Esto aunado a la explicación de cómo usarlo correctamente de la Mtra. María Eugenia.
- Se debe tener un mejor nivel de redacción y organización de la documentación para hacer una tesis.
- Es importante aprender a citar usando APA, y darle crédito a las personas que investigaron artículos previos al sustentante.
- Con herramientas sencillas de utilizar (como lo fue VideoScribe) se puede realizar un material multimedia muy interesante y creativo para hacerlo más llamativo y estimulante.
- Se tiene que redactar la lección aprendida en el momento que se generó, debido a que se olvidan y ya no se registran.
- No aplicar la buena práctica de generar un documento que lleve un control de versiones. Al final de la tesis, se realizó sobre el mismo documento varios formatos para la página web y no se generaron respaldo de ellos, provocando que cuando se tuvieron revisiones surgieron correcciones y se tuvo que realizar todo nuevamente.

Glosario

- **Actividad:** Es un paso o acción que el usuario o software realiza como tarea dada; ésta se representa mediante un rectángulo con las esquinas redondeadas.
- **Algoritmo:** Conjunto de reglas que dan lugar a una serie de pasos/operaciones/acciones para resolver un problema en específico.
- **ArgoUML:** Software libre para diseñar diagramas de actividades.
- **Bifurcación:** Línea de manera horizontal o vertical (dependiendo del diseño del diagrama) la cual ayuda a un mejor entendimiento y comprensión de un diagrama de actividades. Es bastante útil cuando se tienen diferentes opciones.
- **Constante:** Espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato, el cual “no” es modificable a lo largo del algoritmo o programa. Esta constante debe contener una etiqueta o nombre.
- **C#:** Lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft.
- **Diagrama de actividades:** es una representación gráfica del algoritmo o proceso que muestra un flujo de trabajo a través de una serie de pasos.
- **Do-While:** Estructura de control de tipo iteración. Esta estructura funciona de manera similar al While, con la diferencia que el código de este ciclo se ejecutará al menos una vez. De ser verdadera la condición se ejecutará más veces.
- **Estructura de control:** Estructura que permite definir el flujo correcto de un programa mediante la correcta ejecución de instrucciones.
- **For:** Estructura de control de tipo iteración también conocida como estructura cíclica, ya que permite ejecutar una o varias líneas de código de forma repetitiva.
- **If:** Estructura de control de tipo selección que permite la toma de una decisión, en la cual se tendrán dos posibles respuestas, una negativa y una positiva.

- **Lógica computacional:** Es la acción de tomar una decisión, la cual tiene consecuencias, ya sean positivas o negativas.
- **Microsoft:** Empresa informática multinacional fundada en 1975 en Estados Unidos por Bill Gates y Paul Allen.
- **Operadores:** Es un símbolo que tiene una función predefinida, es decir, ya se conoce lo que esta significa y realizará.
- **Plan de configuración:** es un documento formal en el cual se especifican todas la configuraciones de un proyecto y se parametrizan todos los documentos en cuanto al tipo de letra, imágenes, interlineados y justificación del documento. En los diagramas se estandariza el color de los diagramas, el color para las notas, nomenclatura y la herramienta para la elaboración de los mismos. De igual forma, detalla el cronograma de actividades, responsabilidades asignadas mediante los documentos correspondientes, los recursos requeridos y las herramientas.
- **Programación:** Es la acción de ordenar una serie de acciones o pasos para cumplir con cierto objetivo.
- **Pruebas de escritorio:** También conocida como “casos de prueba” establecen el resultado esperado y el método de evaluación del resultado. Se desarrollan para verificar un comportamiento esperado, el cual, al ser correcto, se le denomina “prueba positiva”. Cuando se dan casos donde el comportamiento no es el esperado, se le nombra como “caso de prueba negativo”, ya que se muestra una condición inaceptable, anormal o inadecuada.
- **Rational Rose:** Herramienta de diseño de diagramas de actividades, permite representar gráficamente el sistema, permitiendo hacer énfasis en los detalles más importantes. Es propiedad de IBM.
- **StartUML:** Software libre para diseñar diagramas de actividades.
- **Swimlane:** Área en la cual estarán representadas gráficamente las actividades que le corresponde hacer tanto al usuario como al sistema.

- **Switch:** Estructura de control de tipo selección la cual permite agilizar la toma de decisiones múltiples. Esta estructura analiza casos y no condiciones como el if.
- **Tipo de Dato:** Es la propiedad que puede tomar determinado valor para tipificar qué operaciones se le pueden realizar y cómo es representado este valor por la computadora.
- **Variable:** Espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato, el cual es modificable a lo largo del algoritmo y programa. Esta variable debe contener una etiqueta o nombre.
- **VideoScribe:** Software inglés que crea videos con dibujos escritos a mano y de uso fácil.
- **While:** Estructura de control de tipo iteración la cual funciona como un ciclo de preverificación, en la cual se evalúa una condición, esta, al ser verdadera, ejecutará el código que tendrá inmerso, de lo contrario, no se ejecutará.

Referencias

Abellanas, m. (1991). Análisis de algoritmos y Teoría de grafos (1a. ed.). México: Macrobit.

Arrioja Landa, N. (2010). C# - Guía Total del Programador (1a. ed.). Buenos Aires: Fox Andina; Gradi;Banfield – Lomas de Zamora.

Hernández, M. (2016). Clase 2. Variables y Constantes. Recuperado de umad.duckdns.org:8000/ISF/Analisis/tutAnalisis_01/Opcion1.html

Hernández, M. (2016). Clase 4. Estructuras de Control. Recuperado de umad.duckdns.org:8000/ISF/Analisis/tutAnalisis_01/Opcion1.html

IBM. (s.f.). Rational Enterprice Architect. Recuperado de https://www-03.ibm.com/software/products/es/enterprise

IBM. (s.f.). Rational Rose. Recuperado de https://www-03.ibm.com/software/products/es/ratsadesigner

IBM. UML Activity Diagram. (s.f.). Recuperado el 31 de enero del 2018, de https://www.ibm.com/support/knowledgecenter/SS6RBX_11.4.2/com.ibm.sa.oomethod.doc/topics/c_UML_Activity_diag.html

La Torre. (2017). Actividad 1 - Diagrama de Flujo – Informática. Recuperado de https://sites.google.com/site/abstractionforconstruction/preguntas

Microsoft. Introducción al Lenguaje C# y .NET Framework. (s.f.). Recuperado de https://docs.microsoft.com/es-es/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

Microsoft. Software Profesional. (s.f.). Recuperado de https://www.microsoft.com/es-mx/store/b/software

Salgado, A., Alonso, I., Gorina, A., & Tardo, Y. (2013). Lógica Algorítmica para la Resolución de Problemas de Programación Computacional: Una Propuesta Didáctica. Revista Didasc@Lia: Didáctica Y Educación, 4(1), 67-75.

StartUML. StartUML 2. (s.f.). Recuperado de http://staruml.io/

Time Monitoring Tool, Configuration Management Plan. (2002). 3rd ed. [ebook] Montreal, Quebec: École Polytechnique de Montréal, p.4. Recuperado de http://www.upedu.org/templates/cs/CCM/upedu_ex_cmpln.

Tigris.org. Open Source Software Engeenering Tools. (s.f.). Recuperado de http://argouml.tigris.org/

UPEDU. Checkpoints: Test Case. (s.f.). Recuperado de http://www.upedu.org/process/activity/chklists/ck_tstcs.htm