Trabajo final de Introducción a la Programación

Tema: Raptor

Alumno: Luis Roberto Zúñiga Sánchez.

Profesor: Andrés Jiménez Leandro.

Segundo Cuatrimestre 2016

Contenido

[Resumen ejecutivo: 1](#_Toc458541872)

[Descripción: 2](#_Toc458541873)

[Abstract: 3](#_Toc458541874)

[Objetivo General: 4](#_Toc458541875)

[Objetivo Específico: 5](#_Toc458541876)

[Introduction: 6](#_Toc458541877)

[Desarrollo: 7](#_Toc458541878)

[Conclusiones y recomendaciones: 8](#_Toc458541879)

[Cronograma: 9](#_Toc458541880)

[Bibliografía: 10](#_Toc458541881)

# Resumen ejecutivo:

En este proyecto veremos la función que tiene el diagrama creado en Raptor, el cual clasifica cierta nota en su respectiva categoría, por ejemplo, si está entre 20 y 19 la nota se categoriza en una calificación A, si está entre 18 y 16 será una B.

# Descripción:

Aquí encontraremos la parte teórica de este trabajo, conoceremos más sobre la aplicación y los métodos que se llevan a cabo paso a paso para llegar al resultado final. Vamos a encontrar un abstract con el resumen en inglés para que pueda ser entendido por usuarios sin conocimiento del español, también nos toparemos con los objetivos generales y específicos que llegaremos a lograr con la realización del trabajo y sin dejar de lado los obstáculos y limitaciones que encontramos a la hora de la realización que para corregir y poder avanzar tuvimos que analizar fuertemente el problema y encontrar una solución que testeamos varias veces.

La aplicación Raptor es bastante utilizada para la creación de diagramas la cual ayuda a los estudiantes a visualizar sus algoritmos fácilmente y lo mejor es que Raptor es distribuido libremente como un servicio a la comunidad educativa.

Este tipo de diagramas nos ayuda a resolver problemas fácilmente sin necesidad de hacer un esfuerzo mayor al de ir introduciendo datos, sin embargo, existen diagramas muy complejos los cuales llevan un mayor tiempo de dedicación y mayores testeos o pruebas antes de poder entregar el proyecto como concluido.

# Abstract:

Here we have a summary in English, which we will introduce what quremos achieve with this project.

Raptor is a programming environment based on flowcharts. With this project we want to draw the attention and interest of student programs falicitan creating algorithms and diagrams as they are very rich visual theme and we can see how step by step they are making our processes, which makes them very easy to understand and it is fun to see our results.

RAPTOR has six basic symbols, where each symbol represents a unique type of instruction. The basic symbols are shown at the right. The top four statement types, Assignment, Call, Input, and Output, are explained in this reading, the bottom two types, Selection and Loops, will be explained in a future reading.

INPUT: Allow the user to enter data. Each data value is stored in a variable.

PROCESSING: Change the value of a variable using some type of mathematical calculation.

PROCESSING: Execute a group of instructions defined in the named procedure. In some cases some of the procedure arguments (i.e., variables) will be changed by the procedure's instructions.

OUTPUT: save to a file the value of a variable.

The RAPTOR development environment, allows comments to be added to your program. Comments are used to explain some aspect of a program to a human reader, especially in places where the program code is complex and hard to understand. Comments mean othing to the computer and are not executed. However, if comments are done well, they can make program much easier to understand for a human reader.

The resulting comment can be moved in the RAPTOR window by dragging it, but you typically do not need to move the default location of a comment.

# Objetivo General:

Demostrar lo importante que son los ayudantes para construir diagramas de flujo utilizando específicamente el programa Raptor para de esta forma crear un interés en los compañeros para que utilicen dicho programa y se faciliten la creación de los diagramas a realizar en el futuro.

Dar a conocer la importancia que tienen los programas de control de versiones como lo es GitHub.

# Objetivo Específico:

Realizar una defensa del proyecto final en el cual se explicará paso a paso como se realizó un diagrama con el programa llamado Raptor, de esta manera se enseña lo útil que es efectuar un proceso de procedimientos y verlos ejecutarse uno por uno.

Efectuar una breve explicación de la importancia de GitHub y programas de control de versiones en general para crear la intriga en los alumnos y que amplíen su conocimiento hacia ellos ya que es muy importante conocer de este tema ya que no funcionará mucho como futuros ingenieros en sistemas.

# Introduction:

Este trabajo se crea a partir de un Proyecto propuesto por el profesor, en el cual se debe realizar dos tipos de diagramas en Raptor. El primer diagrama es creado para calificar las notas de los estudiantes en diferentes categorías: A, B, C, D y E, para ellos se le pide al usuario que ingrese una nota y dependiendo en cual rango se encuentre, así será agrupada. El segundo diagrama se crea de la misma manera, sin embargo, este segundo debe llevar lo que se conoce como ciclo While-Do, lo cual hace que el diagrama vuelva a pedir la nota si esta no cumple con algunas especificaciones o muestre el final del diagrama categorizando la nota en su debida agrupación.

# Desarrollo:

En el actual proyecto se debía crear el diagrama que clasifique las notas en alguna categoría como se menciona anteriormente, para ello debíamos analizar cuidadosamente los problemas que se nos van a presentar a la hora de la elaboración y empezar por leer un poco acerca de lo que trataba Raptor y cómo funcionaban sus botones, para que servían cada uno y la manera adecuada de utilizarlos, seguidamente se fueron posicionando los Input para pedirle al usuario una variable la cual almacenamos con nombre “nota”, a continuación se deben ir creando if, el primero contendrá la condición que “nota” sea mayor o igual a 19 o la nota debe ser igual a 20, si esto se cumple entonces la nota se clasifica en una “A” y si no se crea otro if que tendrá la condición que “nota” sea mayor o igual a 16 o la “nota” sea igual a 18, si se cumple entonces la nota se clasifica en una “B” y si no se tendrá que unir otro if que tendrá la condición de que “nota” sea mayor o igual a 13 o la “nota” sea igual a 15, si se cumple entonces se clasifica en una “C” pero si no se cumple se agrega otro if con condición de “nota” sea mayor o igual a 10 o la nota debe ser igual a 12, si se cumple la “nota” se categoriza en una “E”, de lo contrario se clasifica en una “D”. A la hora de crearlo es bastante sencillo, sin embargo, a la hora de probar varias notas llegamos al problema que si se introduce una nota negativa, la cual no existe, se categoriza la “nota” en una “E” por lo que está erróneo, por ello se tuvo que crear un if inicial que nos clasificara la “nota”, si esta es negativa entonces nos botara del programa y si era positiva o igual a cero entonces continuara con normalidad. Al final se llega a otro obstáculo tras analizar el diagrama y se concluye en la necesidad de un bucle While-Do el cual nos ayuda a que el programa no nos bote cuando introducimos una nota menor a cero, esta se hace utilizando dos cuadros de asignación, uno se utiliza al inicio para poner un valor cualquiera a una variable que llamamos “cambio”, en nuestro diagrama manejamos el valor de cualquier numero con excepción de Cero; el segundo cuadro de asignación se posiciona después del Output que nos indica la nota y ese cuadrado se crea para hacer cambiar ese número cualquiera por un “uno” en este caso, así no se cumplirá la primera condición del ciclo y esto nos llevará al final del programa, ya que, de no cumplirse esa condición de un valor distinto de Cero, se crea un ciclo infinito que volverá a preguntar la nota a clasificar.

# Conclusiones y recomendaciones:

Como conclusiones se puede resaltar que el programa utilizado en este trabajo para crear el diagrama de flujo (Raptor) es bastante útil a la hora de realizar dichas tareas y sería bueno que se implemente bastante la utilización de este porque nos ahorraría mucho el proceso de creación de diagramas, ya que es muy sencillo de manejar, con leer previamente el manual de dicho programa ya se es capaz de crear un esquema algo complejo, pero sin numeroso esfuerzo.

En las recomendaciones claramente se puede incluir que se debe realizar un análisis previo a la estructuración del diagrama para saber qué forma se le va a dar a nuestro diagrama de flujo, aunque a la hora de la creación se nos den varias complicaciones que se pueden resolver fácilmente sin tener mucho conocimiento. Se puede aclarar que no se necesita saber sobre códigos o lenguajes de programación para llevar acabo la creación de un diagrama como este que se creó en el trabajo y es bueno intentar hacer cosas nuevas que nos ayudarán de cierta manera para minimizar algún trabajo y perder el miedo hacia los obstáculos que se van dado en todo trabajo, no debemos centrarnos en nuestro fracaso de forma negativa, de lo contrario, se aconseja verlo positivamente para mejorar el diagrama analizando nuestro error.

# Cronograma:

Domingo 17/07/16

5:00pm a 5:25... Realizar diagrama establecido.

5:25pm a 5:58... Efectuar diagrama con el bucle While-Do.

7:20pm a 7:30... Modificar el diagrama para agregar validación con variable menor a Cero.

Lunes 18/07/16

6:10pm a 6:12... Subir el diagrama establecido a GitHub.

Lunes 08/08/16

5:00pm a 5:58pm… Empezar el trabajo escrito.

8:40pm a 8:45pm… Subir Documentación con el inicio del trabajo escrito.

11:13pm a 11:17pm… Modificar el diagrama con Bucle While-Do y cambiar “camb” por “Cambio”

Martes 08/08/16

7:00pm a 9:13pm… Terminar trabajo escrito del proyecto

9:14pm a 9:20pm… Agregar Cronograma al trabajo.

9:20pm a 9:30pm… Terminar paginación y tabla de contenidos.

# Bibliografía:

Información extraída de <http://raptor.martincarlisle.com>