Universiada Autónoma de Yucatán

Facultad de Matemáticas

Programación Estructurada

Ingeniería de Software

Segundo Semestre

Enero-Julio 2018

Proyecto Ajedrez

Integrantes:

* José Manuel Patrón Estrada
* José Manuel Suarez de Regil
* Armando Valdivieso Peralta
* Mauro José Zapata Pantoja

Tabla de contenidos:

Introducción……………………………………………………………………….3

Definición de Requerimientos……………………………………….……3

Casos de uso………………………………………………………………………4

Estándares de Codificación………………………………………………..7

Procesos de Desarrollo………………………………………………………9

**Introducción**

Nuestro equipo de trabajo se plantea la idea de realizar un software que permita a una persona poder analizar partidas de ajedrez que ya se hallan concluido, queremos que en cada movimiento de la partida tenga acceso a estadísticas como el numero de piezas que tiene cada uno, que piezas se perdieron, la visualización del tablero, y en caso de que el usuario desee podría regresar a algún punto de la partida y meter las instrucciones para así ver cuál hubiera sido el resultado si jugaba de otra manera, por lo que el software también servirá como un juego de ajedrez normal.

**Definición de Requerimientos**

En nuestro sistema sólo se tiene un actor que va a ser el usuario que va a revisar o realizar el juego de ajedrez, es un aspecto importante tener en cuenta el hecho de que es el único actor en nuestro sistema pues quiere decir que una vez terminado no podemos realizar cambios o agregar algún tipo de cosa ya sea funcionalidad o información.

Vamos a iniciar con los **requerimientos funcionales** del sistema que se listarán a continuación:

* El sistema debe permitir al usuario escribir la instrucción del movimiento de ajedrez.
* El sistema desplegara el tablero de ajedrez con las piezas en posición en relación con las instrucciones puestas.
* El sistema desplegara una serie de datos como número de piezas en el tablero, turno, número de movimientos después de cada acción.
* El sistema dará la opción al usuario de regresar a cualquier punto en la partida.
* Al momento de finalizar el encuentro se mostrará una pantalla de fin que incluirá datos de la partida y mostrará el ganador.

Ahora para los **requerimientos no funcionales** los dividiremos en dos apartados los que tienen relación a la eficiencia del sistema y los que se relacionan con la usabilidad del sistema, no tenemos requerimientos no funcionales en la parte de seguridad pues creemos no es necesario en nuestro software ya que no maneja información.

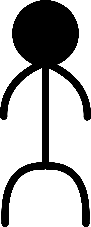
**Eficiencia:**

* El sistema permitirá al usuario ingresar más de 50 instrucciones de ajedrez.
* El sistema debe tardar menos de 10 segundos en imprimir la pantalla con el tablero después de que se ingrese una instrucción.

**Usabilidad:**

* El sistema debe permitir al usuario realizar cualquier acción en menos de 3 clicks.
* En caso de que el usuario cometa un error al ingresar la instrucción el sistema le mostrara un mensaje diciendo como se debe escribir la instrucción.
* El sistema contará con un manual de usuario.
* El sistema debe poseer una interfaz grafica para una mejor visualización de los datos.

**Casos de Uso**



Usuario

Ahora procederemos a describir cada uno de los casos de uso como se explica anteriormente en el texto nuestro único actor es el usuario de tal manera que en los casos de uso el es el único que interactúa.

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | **Ingresar Instrucciones** | |
| Precondición | El usuario debe de saber el formato de ingreso de instrucciones de Ajedrez, así como conocimiento básico del juego | |
| Descripción | El sistema deberá realizar los pasos que se escriben a continuación cuando el usuario ingrese instrucciones. | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario ingresa la instrucción y le da click a siguiente. |
| 2 | El sistema revisa la instrucción y acepta que este bien escrita. |
| 3 | El sistema despliega el tablero con las piezas en las nuevas posiciones y los datos de juego actualizado. |
| 4 | En caso de que se gane la partida en ese movimiento el sistema despliega una pantalla diferente con los datos de Victoria. |
| Postcondición | El sistema se queda mostrando los datos del último movimiento hasta que el usuario ejecute alguna acción. | |
| Excepciones | Paso | Acción |
| 2 | El sistema revisa la instrucción y detecta que está mal escrita. |
| 3 | El sistema despliega un mensaje que informa al usuario del error. |
| Comentarios | La manera en la que se debe de escribir la instrucción vendrá en el manual de usuario que tendrá el sistema. | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | **Regresar a un momento en la partida** | |
| Precondición | Haber realizado dos o más movimientos. | |
| Descripción | El sistema deberá regresar al momento de partida que el usuario haya seleccionado con el número de movimiento. | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario ingresa el número de movimiento al cuál regresar. |
| 2 | El sistema revisa el número y lo toma como valido. |
| 3 | El sistema despliega el tablero con las piezas en las posiciones y los datos de juego en la posición pedida |
| Postcondición | El sistema elimina todos los datos posteriores a este movimiento esperando por nuevas instrucciones. | |
| Excepciones | Paso | Acción |
| 2 | El sistema revisa el número y lo toma como no valido |
| 3 | El sistema despliega un mensaje que informa al usuario del error. |
| Comentarios | Durante la partida en los datos aparece el número de movimiento para que el usuario pueda llevar un control. | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | **Reiniciar el partido** | |
| Precondición | Haber realizado al menos un movimiento. | |
| Descripción | El sistema deberá eliminar todos los datos de la partida y regresar a la pantalla 1 para que el usuario pueda ingresar las instrucciones. | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario ingresa el comando para reiniciar la partida |
| 2 | El sistema revisa el comando y lo toma como valido. |
| 3 | El sistema la primera pantalla. |
| Postcondición | El sistema elimina todos los datos y espera a una nueva instrucción. | |
| Excepciones | Paso | Acción |
| 2 | El sistema revisa el comando y lo toma como erróneo |
| 3 | El sistema despliega un mensaje que informa al usuario del error. |
| Comentarios | El comando estará escrito en el manual de usuario que tendrá el sistema. | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | **Finalizar** | |
| Precondición | Haber realizado al menos un movimiento. | |
| Descripción | El sistema mostrara la pantalla de Empate. | |
| Secuencia normal | Paso | Acción |
| 1 | El usuario ingresa el comando para finalizar la partida. |
| 2 | El sistema revisa el comando y lo toma como valido. |
| 3 | El sistema despliega la pantalla de Empate con los datos de la partida |
| Postcondición | El sistema muestra la pantalla y espera una nueva acción. | |
| Excepciones | Paso | Acción |
| 2 | El sistema revisa el comando y lo toma como no valido |
| 3 | El sistema despliega un mensaje que informa al usuario del error. |
| Comentarios | Se muestra una pantalla de empate pues finalizar la partida con este comando que estará en el manual de usuario no representa una victoria, se espera que se use en casos en los que se termine en Tablas. (Termino del ajedrez para referirse a empate). | |

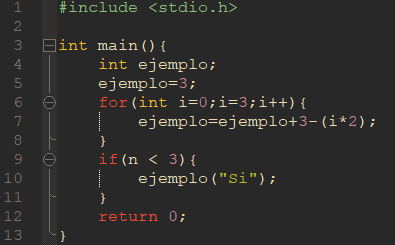
**Estándares de Codificación**

Para poder explicar el estándar de codificación que vamos a usar vamos a dividirlo en partes para hacerlo más sencillo de entender, iniciaremos primero con la estructura del programa, quiere decir todas las instrucciones que se escriban dentro y fuera de cualquier función.

**Estructura del Programa**

Al final de la lista se pondrá una imagen de ejemplo para mayor claridad.

* Siempre se usarán llaves después de cualquier estructura de control
* No se escribirá código justo después de una llave de apertura.
* Sólo se insertará una instrucción por línea.
* Se usarán paréntesis para la jerarquía de operaciones.
* Los operadores binarios serán separados con un espacio por ambos lados.
* Las bibliotecas se escribirán entre angle brackets “< >”.



Ahora se definirán como se nombrarán las variables y las funciones se enlistarán de manera similar a la de Estructura de Programa.

**Variables y Funciones.**

* No se usarán variables globales.
* Las variables que se utilicen como índice de iteración empezaran con la i y seguirán j, k, l hasta que sea necesario.
* Su nombrado será en base a su función
* Se usará el formato camelCase para el caso de variables y funciones de nombre compuesto. Por ejemplo, si tenemos una variable que almacenará el resultado de una suma, podría llamarse con formato camelCase como sumaDigitos.
* Los valores constantes deberán declararse al comienzo del programa usando #define y su nombre deberá escribirse en mayúsculas.
* Para las variables tipo puntero siempre iniciaran su nombre con “ptr”.

Por último, vamos a definir la manera que se realizarán los comentarios.

**Comentarios**

* Los comentarios iniciarán con “/\*” y terminarán con “\*/”.
* Cada bloque de código comenzará con un bloque de comentarios propio en el que al menos se detalle su propósito, los parámetros de entrada que use, posibles valores de retorno, códigos de error o retorno que genere, etc. La parte de los parámetros de entrada y salida estarán bien separados.
* En caso de referirnos a una variable dentro del bloque de comentarios, usaremos comillas simples. Por ejemplo: /\* La variable ’sumaItems’ representa el sumatorio total de ‘Items’ en el array \*/
* Incluiremos un espacio en blanco después de la apertura del bloque de comentarios y antes del cierre.
* Los comentarios serán lo más concisos posibles.

**Proceso de Desarrollo**

Nuestro sistema será codificado en el lenguaje C, para la parte de la interfaz gráfica donde podrá interactuar el usuario utilizaremos la biblioteca GTK ( <gtk/gtk.h >), una herramienta que facilitará el uso de la biblioteca GTK será la herramienta de desarrollo visual Glade.

El equipo utilizará la aplicación discord como medio de comunicación. Para monitorear el avance y llevar un seguimiento de el desarrollo de nuestro programa donde todo el equipo tiene acceso utilizaremos la herramienta Github.

Se va a llevar un control de acuerdo a cuanto aporte cada integrante del equipo al proyecto.