|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería en Software | **GUÍA**  No. 01 | **TIEMPO ESTIMADO:**  2 semanas |
| **ASIGNATURA:**  Estructura de Datos | **FECHA DE ELABORACION: 27/01/2020**  **SEMESTRE**: Septiembre 2019 – Febrero 2020 | |
| **TÍTULO:**  Indiana Croft | **DOCENTE:** Fernando Solís | |

**OBJETIVO**

1. Realizar un proyecto que incluya funciones recursivas y genere una solución al problema planteado.

**INSTRUCCIONES**

1. Utilice como material principal, aquel indicado en clase por el docente.
2. Utilice información consultada en Internet y conocimiento adquirido en clase.

**ACTIVIDADES**

1. **Ubicación de recursos**
2. Formar grupos de máximo 2 personas.
3. Utilizar la herramienta que desee para el desarrollo del proyecto.
4. **Planteamiento del problema**

Realizar un juego que se autocomplete mediante el uso de funciones recursivas. El juego consistirá en un laberinto de N\*N el cual consista en pisar cada uno de los cuadrados que se muestra en pantalla tantas veces como el numero que se muestre en el cuadrado, es decir si este tiene un numero 3 encima deberá cruzar 3 veces por el mismo cuadrado, y para ganar el juego todos los cuadrados deben quedar en 0 completando asi el laberinto. En un documento aparte se generara la solución paso por paso.

1. **Entregable (s)**
2. El proyecto funcionando correctamente y sin errores.
3. El código del proyecto para la revisión respectiva.
4. El modelado del proyecto con cada una de las clases y métodos que se utilizaron.

**MARCO TEÓRICO**

**TDA**

Un TAD se define como una estructura algebraica compuesta por un conjunto de objetos abstractos que modelan elementos del mundo real, y un conjunto de operaciones para su manipulación.

Un TAD es un ente cerrado y autosuficiente, que no requiere de un contexto específico para que pueda ser utilizado en un programa. Esto garantiza portabilidad y reutilización del software.

Las partes que forman un TAD son:

* 1. atributos (tipos de datos, identificadores, etc.)
  2. funciones (rutinas) que definen las operaciones válidas para manipular los datos (atributos).

A la forma de operar y encerrar los atributos y funciones dentro un TAD se le denomina encapsulamiento.

Para implementar un TAD se siguen dos pasos:

1. Diseñar las estructuras de datos que van a representar cada objeto abstracto.
2. Desarrollar una función, por cada operación del TAD, que simule el comportamiento del objeto abstracto, sobre las estructuras de datos seleccionadas.

Las operaciones de un TAD se clasifican en 3 grupos, según su función sobre el objeto abstracto:

* 1. ***Constructora***: es la operación encargada de crear elementos del TAD.
  2. ***Modificadora***: es la operación que puede alterar el estado de un elemento de un TAD.
  3. ***Analizadora***: es una operación que no altera el estado del objeto, sino que tiene como misión consultar su estado y retornar algún tipo de información.

**FUNCIONES RECURSIVAS**

En [lógica matemática](https://es.wikipedia.org/wiki/L%C3%B3gica_matem%C3%A1tica) y [computación](https://es.wikipedia.org/wiki/Computaci%C3%B3n), las funciones recursivas o también conocidas como funciones recursivas-μ son una clase de [funciones](https://es.wikipedia.org/wiki/Funci%C3%B3n_matem%C3%A1tica) de los [números naturales](https://es.wikipedia.org/wiki/N%C3%BAmero_natural) en los números naturales que son «computables» en un sentido intuitivo. De hecho, en [teoría de la computabilidad](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computabilidad) se demuestra que las funciones recursivas son precisamente las funciones que pueden ser calculadas con el formalismo de cómputo más general conocido como lo son las [máquinas de Turing](https://es.wikipedia.org/wiki/M%C3%A1quina_de_Turing).

Las funciones recursivas están relacionadas con las [funciones primitivas recursivas](https://es.wikipedia.org/wiki/Recursi%C3%B3n_primitiva) y su [definición inductiva](https://es.wikipedia.org/wiki/Funci%C3%B3n_recursiva#definici%C3%B3n) se construye basándose en la de las funciones primitivas recursivas (estas se obtienen por medio de recursión primitiva y composición de [funciones iniciales](https://es.wikipedia.org/w/index.php?title=Funciones_iniciales&action=edit&redlink=1)). No toda función recursiva es primitiva recursiva. El ejemplo más conocido es la [función de Ackermann](https://es.wikipedia.org/wiki/Funci%C3%B3n_de_Ackermann).

**CONCLUSIONES**

Con este proyecto se aprendieron varias cosas que ayudaron a mejorarlo como el uso de funciones recursivas para la solución de problemas, aumento de la eficacia en la búsqueda de datos entre otras aplicaciones que se le pueden dar a las funciones recursivas. El uso de Tipos de Datos Abstractos también ayuda a mejorar el código que se implementó para la solución de este proyecto, siendo este más entendible y “limpio” para la mejor comprensión al momento de realizar la revisión del proyecto.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DOCENTE RESPONSABLE COORDINADOR DE ÁREA

Ing. Fernando Solis. MsC. PhD. Rodrigo Fonseca.