

Curso: Estructuras de Datos 1

Proyecto 2 (Proyecto Grupal)

Nombre proyecto: Snake The Game

Integrantes:

Amanda Campos Zeledón

Carlos Melegatti Sánchez

Profesor: Christian Sibaja Fernández

Periodo: Tercer Cuatrimestre 2017

Fecha de entrega: 11 de diciembre del 2017

TAD SnakeList:

1. Representación abstracta:



1. Determinar el invariante:

Los valores que se emplean para la posición “X” y la posición “Y” (posición que representa un punto sobre el eje de las coordenadas X y Y respectivamente), deberán ser valores enteros positivos y/o negativos.

1. Definir las operaciones validas:

NodoListaSnake \* getCabeza() -> int // Analizadora / Retorna el valor del puntero al nodo cabeza.

setCabeza(NodoListaSnake\*) -> void // Modificadora / Modifica el valor del puntero al nodo cabeza.

insertar() -> void // Modificadora / Agrega un nuevo nodo a la estructura de datos “SnakeList”.

move(int, int); -> void // Modificadora / Modifica los valores de los ejes X y Y para gestionar el movimiento de la cabeza de la serpiente.

1. Manejo del error:

Se mostrará el nombre del usuario, el puntaje y la cantidad de intentos disponibles en la partida de cada uno de los jugadores en el momento en que se ingresen los datos correctos de los usuarios que jugaran el juego.

TAD ColaJugadores:

1. Representación abstracta:



1. Determinar el invariante:

Los valores que se emplean en el TAD ColaJugadores son “cabeza, cola y Jugador (valores de tipo Jugador)”, por lo que estos deberán ser ingresados y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

setCabeza(NodoColaJugadores \*) -> void // Modificadora / Modifica el valor del puntero al nodo NodoColaJugadores

NodoColaJugadores\* getCabeza(void); -> int // Analizadora / Obtiene la información del puntero al nodo NodoColaJugadores

void setCola(NodoColaJugadores\*); -> void // Modificadora / Modifica la información del nodo NodoColaJugadores,

NodoColaJugadores \* getCola(); Obtiene la información del puntero al nodo NodoColaJugadores

agregarJugador(Jugador \*);-> void // Modificadora / Agrega un jugador de tipo Jugador a la lista de jugadores.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

TAD PilaFrutas:

1. Representación abstracta:



1. Determinar el invariante:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

Pilafrutas();-> void // Constructura // Crea la TAD PilaFrutas

~Pilafrutas();-> void // Destructora // Destruye la TAD PilaFrutas

NodoPilaFrutas \* getTope();-> void // Consultora / Extrae el tope de la pila PilaFrutas.

setTope(NodoPilaFrutas \*);-> void // Modificadora / Modifica el tope de la pila.

pushElem(string); -> void // Modificadora / Modifica el elemento en la lista PilaFrutas.

NodoPilaFrutas\* popElem();-> int // Extractora / Extrae el elemento del NodoPilaFrutas.

int getLongitud();-> void // Consultora / Extrae el valor de la longitud de la pila PilaFrutas.

setLongitud(int); -> void // Modificadora / Modifica el valor de la longitud de la pila PilaFrutas.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos. El sistema, deberá validar que los tipos de datos sean los correctos.

TAD ListaMejoresPuntajes:

1. Representación abstracta:



1. Determinar el invariante:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

ListaMejoresPuntajes();-> void // Constructura / Crea la TAD ListaMejoresPuntajes.

~ListaMejoresPuntajes();-> void // Destructora / Destruye la TAD ListaMejoresPuntajes.

NodoListaMejoresPuntajes \* getCabeza();-> // Consultora / Extrae el valor de la cabeza del nodo NodoListaMejoresPuntajes.

setCabeza(NodoListaMejoresPuntajes \*);-> void // Modificadora / Modifica el valor de la cabeza del nodo NodoListaMejoresPuntajes.

mostrarMejoresPuntajes();-> void // Consultora / Extrae los valores contenidos en la lista ListaMejoresPuntajes.

insertarOrdenado(int); -> void // Modificadora / Modifica el contenido de la lista ListaMejoresPuntajes.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos. El sistema, deberá validar que los tipos de datos sean los correctos.

TAD Jugador:

1. Representación abstracta:



1. Determinar el invariante:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

Jugador();-> void // Constructora / Crea la TAD Jugador.

~Jugador();-> void // Destructora / Destruye la TAD Jugador.

getNombre();-> string // Consultora / Extrae el nombre de la lista Jugador.

setNombre(string); -> void // Modificadora / Modifica la información el nombre de la lista Jugador.

getPos();-> int // Consultora / Extrae la informacion de la posición del jugador de la lista Jugador.

setPos(int); -> void // Modificadora / Modifica la posición del jugador de la lista Jugador.

getPuntaje();-> int // Consultora / Extrae la información del puntaje del jugador de la lista Jugador.

setPuntaje(int); -> void // Modificadora / Modifica el puntaje del jugador de la lista Jugador.

getVidas();-> int // Consultora / Extrae la cantidad de intentos disponibles de la lista Jugador.

void setVidas(int); -> void // Modificadora / Modifica la cantidad de intentos disponibles para cada jugador.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos. El sistema, deberá validar que los tipos de datos sean los correctos.

TAD Tail:

1. Representación abstracta:



1. Determinar el invariante:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

Tail();-> void // Creadora / Crea la TAD Tail.

~Tail();-> void // Destructora / Destruye la TAD Tail.

getY();-> int // Consultora / Extrae la información del eje Y de cola Tail.

setY(int); -> void // Modificadora / Modifica la información del eje Y de la cola Tail.

getX();-> int // Consultora / Extrae la información del eje X de cola Tail.

setX(int); -> void // Modificadora / Modifica la información del eje X de la cola Tail.

getTail();-> string // Consultora / Extrae la información de la cola Tail.

setTail();-> void // Modificadora / Modifica la información de la cola Tail.

setTail(string); -> void // Modificadora / Modifica la información de la cola Tail.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos. El sistema, deberá validar que los tipos de datos sean los correctos.

TAD Gestor:

1. Representación abstracta:



1. Determinar el invariante:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos.

1. Definir las operaciones validas:

Gestor();-> void // Creadora / Crea el gestor.

~Gestor();-> void // Destruye el gestor.

begin();-> void // Inicia la ejecución del sistema.

crearJugadores(int x); -> void // Llamado al método para crear jugadores.

Jugador \* numeroJugador(int); -> void // Llamado al método extraer el número de jugador.

start(int); -> void // Llamado al método para ejecutarlo.

inicializarGestor();-> void // Llamado al método para inicializar el gestor.

draw(int); -> void // Llamado al método para iniciar a dibujar.

datosJugador(int); -> void // Llamado al método para ejecutarlo.

drawWidth();-> void // Llamado al método para dibujar las paredes laterales.

drawWalls();-> void // Llamado al método para dibujar las paredes.

drawSnake(int i, int j); -> void // Llamado al método para dibujar la snake.

drawFruit(int i, int j); -> void // Llamado al método para dibujar la fruta.

setFruit();-> void // Llamado al método para ejecutarlo.

setFruits();-> void // Llamado al método para ejecutarlo.

setUp();-> void // Llamado al método para ejecutar la configuración inicial del juego.

logic(Jugador\*);-> void // Llamado al método para ejecutar la lógica del juego.

input();-> void // Llamado al método para ejecutar los comandos de captura del juego.

1. Manejo del error:

Todos los datos se deberán ingresar y el sistema requiere valores válidos para cada uno de ellos. El sistema, deberá validar que los tipos de datos sean los correctos.