Mamani Pérez Julieta Agustina

Trabajo de INtegrador

Orientación Informática - Estructura de Datos (2019)

ESTRUCTURAS DE DATOS ELEGIDAS

Constantes*: const int MAX\_J=50, MAX\_T=28;*

MAX\_J hace referencia a la máxima cantidad de jugadores que pueden ser registrados en el sistema y MAX\_T a la cantidad de casillas que tiene un tablero de juego.

Estructuras para realizar lista simple (contendrá a los jugadores)

tipo puntero que permite referenciar registros tnodo

*typedef struct tnodo \*pnodo;*

TDA Jugador, se establece los campos que debe tener un jugador

*typedef struct Jugador {*

*string nombre; string apellido; string email; int puntaje;*

*} jugador;*

Declaración del nodo de la lista tlista (donde se almacenará a los jugadores), donde el campo dato es del tipo Jugador debido a que voy a almacenar un jugador en cada nodo.

*typedef struct tnodo {*

*pnodo sig; Jugador dato;*

*} tnodo;*

Declaración de la lista con un contador para saber la cantidad de jugadores cargados y no sobrepasar el límite MAX\_J

*typedef struct tlista {*

*int cant;*

*pnodo i;*

*} tlista;*

Estructuras para realizar lista doble (contendrá casillas para un tablero)

tipo puntero que permite referenciar registros tcasilla

typedef struct tcasilla \*pcasilla;

TDA casilla, determina los datos que tiene una casilla, el numero de casilla y el suceso.

*typedef struct casilla {*

*int dato; //número de casilla en el tablero*

*string suceso;*

*} casilla;*

Declaración del nodo de la lista doble Tablero (que será utilizada para generar el tablero), donde el campo dato es del tipo casilla.

*typedef struct tcasilla {*

*pcasilla sig;*

*pcasilla ant;*

*casilla dato;*

*} tcasilla;*

Declaración de la lista Tablero que tendrá un puntero al inicio y al final y además contará con un contador para no sobrepasar el limite de casillas que es igual MAX\_C.

*typedef struct Tablero {*

*int cant;*

*pcasilla i;*

*pcasilla f;*

*} Tablero;*

Estructuras para realizar árbol binario (contendrá los datos de los jugadores con mejor puntaje)

Tipo de puntero que permite referenciar registros tArbol

*typedef struct tArbol \*pArbol;*

Declaración del nodo que formara el árbol binario de búsqueda utilizado para determinar el ranking. Como va a ser utilizado en un árbol binario se declara dos campos del tipo pArbol (izq= izquierda y der=derecha), además de un campo dato, que llevara los datos del nodo, este caso es del tipo Jugador porque debe guardar la información de un jugador para luego mostrarla. Pero para las operaciones solo se tomará en cuanta el campo Jugador.puntaje.

*typedef struct tArbol {*

*Jugador dato;*

*pArbol izq;*

*pArbol der;*

*} tArbol;*

bool lista\_vacia(tlista jugadores){

return (jugadores.i=NULL);

}

Operaciones básicas modificadas

* Iniciar lista
* Crear nodo
* Agregar nodo: agregar\_final
* Mostrar (recorrido de la lista)
* Buscar (busca un valor en la lista)
* Iniciar árbol
* Recorrido: Pre-orden (RAÍZ-izquierda-derecha)
* Búsqueda (árboles binarios de búsqueda)
* Inserción (árboles binarios de búsqueda)
* Eliminación (árboles binarios de búsqueda

El procedimiento void iniciar\_lista (tlista &jugadores) es la misma lógica que la operación fundamental Iniciar\_lista de una lista con contador. Void iniciar\_tablero()

void listarTodos (tlista jugadores) Se construye a partir de la modificación de la operación básica Mostrar, en la operación se listarán todos los jugadores registrados.

El procedimiento bool existe(tlista jugadores, string email) busca un valor en la lista y si este existe devuelve un valor falso.

El procedimiento void consultar(tlista jugadores, string email)busca un valor en la lista y lo muestra por pantalla. Se basa en la operación fundamental Buscar().

El procedimiento void modificar(string email, tlista &jugadores)

El procedimiento void registrar\_J(tlista &jugadores) crea un nodo y lo agrega al final de la lista

El juego debe presentar un menú de opciones que permita:

a) Administrar jugadores

b) Generar tablero de juego

c) Iniciar la partida

d) Ranking jugadores

a- Administrar jugadores (sub-menú) permite registrar jugadores (nombre, apellido, email y puntaje), consultar/modificar un jugador específico y listar todos los jugadores. No está permitido registrar jugadores repetidos, y podrá almacenarse como máximo hasta 50 jugadores.

b- Generar tablero crea los 28 casilleros del viaje y asigna a éstos aliens, tormentas, asteroides y estrellas. El casillero 28 es la LUNA.

c- La partida sólo podrá iniciar si se creó el tablero y existen como mínimo 2 jugadores registrados. Los jugadores de la partida se eligen antes de comenzar a jugar.

d- El ranking de jugadores mostrará los 3 jugadores con los mejores puntajes. Este ranking se generará usando un árbol binario de búsqueda.

Los datos de los jugadores se almacenarán en listas simples, mientras que el tablero de juego deberá construirse con listas dobles.

En cada partida, y de forma aleatoria, se asignarán casilleros para:

• 2 estrellas (2)

• 3 aliens (3)

• 3 tormentas solares (4)

• 3 lluvias de asteroides (5)

|  |  |
| --- | --- |
| 2 |  |
| 2 |  |
| 3 |  |
| 3 |  |
| 3 |  |
| 4 |  |
| 4 |  |
| 4 |  |
| 5 |  |
| 5 |  |
| 5 |  |

Las estrellas ocuparán cualquier par de casilleros del tablero, mientras que los aliens, tormentas solares y lluvias de asteroides se asignarán a casillas consecutivas.

Por ej.: en una partida podrían asignarse los casilleros 11, 12 y 13 a aliens. Un mismo casillero NO puede tener asignado más de un evento (por ejemplo, aliens y estrellas). Conforme las naves realicen el viaje, cada casilla visitada aportará 3 puntos al jugador. No obstante, las tormentas solares o las lluvias de asteroides restarán 2 puntos, mientras que las casillas de estrellas sumarán 5 puntos y los casilleros de aliens reducirán el puntaje a 0. Sólo el ganador sumará el puntaje de la partida. El juego debe contener los controles necesarios para ejecutar las opciones sólo cuando se disponga de los datos necesarios.