Pilas

(Clase---------------------------->>>>>

Almacena y recupera datos siguiendo a un orden estrcito. Son estrucutruas LIFO o FILO. Solo se puede acceder por un únic lugar de la pila (en caso de memoria dinámica).

Sigueindo la lógica de las pilas, estas solopueden agregar o elimnar un elemento desde el tope o cima de esta.

Puede implementase guardando los datso en un array (memeoria estática), aunqu también puede usarse un VECTOR. Otra forma sería creando una lista enlazada(memoria dinámica), donde cada elemento representa un Nodo, y el límite de los datos lo determina la memoria del ordenador.

En el TDA los datos almacenados pueden ser de cualquier tipo,

<<<<<<<----------------------------------------------

**Definiciones:**

Subprogramas: Se les nombra así cuando un programa se divide en dos o más partes, los cuales pueden clasificarse en funciones (devuelven un valor) o en procedimientos.

**Formas de implementación de las pilas hay dos**

Memoria Estática: Usando una tabla unidimensional (Arrays)

Memoria dinámica: Mediante el uso de punteros

Aplicaciones de las pilas:

Subprogramas: Al momento de llamar a algún subprograma, módulo o función, podemos usar una pila para guardar el estado de una variable, para después ser recuperados y continuar la ejecución del programa en un punto en el que fue interrumpido (¿Recursividad?), o también sus instrucciones pendientes de ejecución.

También se puede recuperar las direcciones del programa en la que se hizo la llamada.

Ejemplo:

Llamada a un subprograma:

* Si se tiene el programa principal “*Main*”, y llama a los subprogramas *Uno* y *Dos*
* A su vez, el subprograma *Dos,* llama al subprograma *Tres*.
* Al terminar la ejecución de uno de los subprogramas, se regresa al nivel superior inmediato, osea, el que invocó a dicho método que concluyó.

*Main*

*Dos*

*Uno*

*Tres*

Recursividad: En un ejercicio de recursividad, una pila puede ser implementada de manera que guarde las instrucciones a ejecutar con sus respectivas variables en una pila.

Cuando se termina la ejecución se llega al estado básico, se toma la instrucción del tope de la pila y se continúa ejecutando hasta que la pila queda vacía.

Otros usos…

Evaluación de expresiones en notación postfija

**Colas**

**Características importantes:**

* Los elementos de la estructura serán del mismo tipo
* La inserción de información es por el final de la cola

**Algunas aplicaciones de las colas:**

* Desarrollo web: Cunado x usuario hace click en una página web los datos que se producen son de manera rápida, pero los consumiremos de manera más lenta más adelante.

**Colas de prioridad en Java:**

Habrá un atributo asignado al elemento insertado que nos indicará su grado de prioridad, y la inserción de elementos será arbitraria. Un ejemplo de su uso en la computación, sería en la ejecución de un proceso para decidir a cuál de ellos se le asignará la prioridad para ser ejecutado por la CPU (política de asignación de la CPU mediante la política de shirtest-job-first, que minimiza los tiempos de espera de todos los procesos).

Al criterio de comparación de las colas de prioridad se les llama “***orden total”,* y dichas colas se dividen en colas con prioridad de mínimos o prioridad de máximos**

**-Implementación de las colas**

Encontramos útiles dichas colas implementándolas en listas ordenadas, árboles equilibrados e incluso montículos

**Cola de prioridad con montículos binarios:**

Montículo binario (minimal): Permite obtener el mínimo elemento de una lista independientemente del tamaño de esta. Ejemplos de utilidad: Atender a un paciente en un hospital.

Debe cumplir dos propiedades:

1.- Es un árbol binario completo: Todos sus niveles están completos, excepto tal vez el último.

2.- Para cada nodo X con Padre P, el dato P es menor o igual al dato X, el dato del Padre nunca es mayor que el de los hijos y a puede pasar que el nodo raíz no tenga padre.

Para almacenar datos empleamos un array, y los metemos indicados por niveles. Para encontrar datos usamos una expresión matemática como la siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La manera clásica de implementar una cola de prioridad es usar una estructura de datos llamada **montículo binario**. Un montículo binario nos permitirá tanto agregar como hacer avanzar ítems

El montículo binario tiene dos variantes comunes: el **montículo mín**, en el que la clave más pequeña está siempre en el frente, y el **montículo máx**, en el que el valor clave más grande siempre está en el frente. Se implementa la naturaleza logarítmica del árbol binario para representar nuestro montículo. Para garantizar un rendimiento logarítmico, debemos mantener a nuestro árbol equilibrado. (tiene aproximadamente el mismo número de nodos en los subárboles izquierdo y derecho de la raíz).

**Another Fast and Simple DEM Depression-Filling Algorithm Based on Priority Queue Structure**

Some depression cells with heights lower than their surrounding cells may often be found in Grid-based digital elevation models (DEM) dataset due to sampling errors. The depression-filling algorithm presented by Planchon and Darboux works very quickly compared to other published methods. Despite its simplicity and delicacy, this algorithm remains difficult to understand due to its three complex subroutines and its recursive execution. Another fast algorithm is presented in this article. The main idea of this new algorithm is as follows: first, the DEM dataset is viewed as an island and the outer space as an ocean; when the ocean level increases, the DEM cells on the island’s boundary will be inundated; when a cell is inundated for the first time, its elevation is increased to the ocean level at that moment; after the ocean has inun-dated the entire DEM, all of the depressions are filled. The depression-removing processing is performed using a priority queue. Theoretically, this new algorithm is a fast algorithm despite the fact that it runs more slowly than Planchon and Darboux's method. Its time-complexity in both the worst case and in an average case is O(8nlog2(m)), which is close to O(n). The running speed of this algorithm depends mainly on the insertion operation of the priority queue. As shown by the tests, the depression-filling effects of this algorithm are correct and valid, and the overall time consumption of this algorithm is less than twice the time consumed by Planchon & Darboux's method for handling a DEM smaller than 25002500 cells. More importantly, this new algorithm is simpler and easier to understand than Planchon and Darboux's method. This advantage allows the correct program code to be written quickly.