2024-1

UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

MATERIA

ESTRUCTURA DE DATOS

TEMA

PILA

PROFESOR

Mtro. ISC. MIGUEL ANGEL SÁNCHEZ HERNÁNDEZ

Ing. Miguel Angel Sánchez Hernández

Introducción

En la práctica anterior analizamos como se pueden crear la lista, ocupando el enlace de nodos a través de la programación orientada de objetos. Ahora nos toca ver un tipo especial de lista que es la Pila.

Pila

La Pila es un tipo especial de lista (restringida), esto es debido a la colocación de sus elementos que solo se realizan en la cabeza, y la extracción de los mismos es por la cabeza.

Si nos imaginamos a la pila como unas charolas en la cafetería, si llegan las charolas nuevas estas se ponen en la parte de arriba, y cuando se quiere tomar una de ellas, se toma la que está arriba de todas.

Dado esta situación la pila es una estructura de tipo LIFO (last in, first out, último en entrar, primero en salir).

En una pila se pueden tomar elementos solo si la pila los tiene, y se pueden añadir elementos solo si tiene el suficiente espacio.

Tomando esto en cuenta las operaciones de la pila se definen base estos conceptos:

- borrar(): borrar la pila
- estaVacia(): verificar si la pila esta vacía
- insertar(objeto ob): insertar un elemento en la parte superior de la pila
- extraer(): extrae el elemento superior de la pila
- elementoSuperior(): regresa el elemento superior de la pila sin quitarlo

Ejercicio: De acuerdo a la implementación de lista simple de la práctica anterior adapta el código para ocuparlo como una pila.

Problema

En general la pila es útil cuando los datos deben almacenarse y luego recuperarse en orden inverso. Por ejemplo si nosotros queremos realizar un programa que sirva para detectar delimitadores, como sabemos todo programa no es correcto si los delimitadores que ocupa no tienen su pareja. En java tenemos los siguientes delimitadores “(“, “)”, “{“, “}”, “[“, “]” y los de comentarios “/*” y “*/”.

Tomando esto en cuanta entonces debemos leer un carácter si este es un delimitador de apertura entonces se almacena en la pila, si se encuentra un delimitador de cierre en las lecturas siguientes,

se compara con el delimitador de la pila si son iguales se saca de la pila el delimitador de apertura, en caso contrario que no sean iguales, el procedimiento se tiene que cancelar.

Si al terminar la lectura completa del flujo de caracteres, y la pila está vacía entonces se toma como correcto el uso de los limitadores.

Ejercicio 1: Implementar el problema planteado anteriormente ocupando una pila.

Aplicaciones de la Pila

Nosotros sabemos que los operadores matemáticos son los siguientes (+,-,*,/) y los operandos son los que realizamos dichas operaciones. Existen diferentes maneras para representar las operaciones las cuales son:

A + B interfija

+AB prefija

AB- posfija

En la siguiente tabla se muestra unos ejemplos de cómo representar dichas operaciones en interfija, prefija y posfija.

Interfija	Prefija	Posfija
A+B	+AB	AB+
(A+B)*(C-D)	*+AB-CD	AB+CD-*
A-B/(C*D+E)	-A/B*C+DE	ABCD+*/-

Como evaluar una expresión posfija ocupando una pila

El siguiente algoritmo nos da una idea como poder evaluar una expresión posfija

```
Pila=pila vacía;
While (no sea fin de cadena){
        símbolo=siguiente carácter de entrada
        if(símbolo es un operando)
                añadirPila(pila,simbolo)
        else
                operando1=sacarPila();
                operando2=sacarPila();
                valor=resultado de aplicar el símbolo a perando1 y operando 2
                añadirPila(pila,valor)
        end if
}
```

Ejercicio 2: implementar el algoritmo anterior ocupando la clase general de pila que se realizo en la práctica anterior.

Conversión de una expresión interfija a postfija

Ahora se muestra otro bosquejo para convertir una expresión interfija a posfija

```
pila=pila vacía
while(no se fin de la cadena){
        símbolo=siguiente carácter de entrada
        if(símbolo es un operando )
                agregar símbolo a la cadena
        else
                while(no es vacia pila && precedencia(verPila(pila),simbolo)){
                        simboloArriba=extraerPila(pila);
                        agregar simboloArriba a la cadena
                }
                añadir(pila,símbolo)
        end if
}
/* Operadores restantes*/
While(no se fin pila){
simboloArriba=extraerPila(pila)
agregar simboloArriba a la cadena
}
```

Nota: el método de procedencia evalúa la procedencia de los operadores por ejemplo procedencia(*,+) es true, pero procedencia(+,*) es falsa, procedencia(+,+) es true. Este algoritmo no toma los paréntesis.

Ejercicio 3-a: Implementa el algoritmo anterior
Ejercicio 3-b: Que tienes que agregar demás al algoritmo anterior para que también pueda ocupar los paréntesis.

Ejercicio 4: Escribe un programa que lea una cadena de caracteres y determine si es un palíndromo (es una secuencia de caracteres que se lee igual hacia adelante que hacia atrás), ocupa la estructura de pilas para resolverlo.

Ejercicio 5: Una manera de encriptar mensajes(no muy segura), es colocar paréntesis de manera arbitraria, y todo lo que está dentro de un paréntesis, ponerlo al revés, por ejemplo "Olimpiada de Informática" se puede encriptar como "Olimpia(ad) de I(rofn)matica", los paréntesis también se pueden anidar, es decir, otra forma de encriptar el mismo mensaje podría ser "Olimpia(am(nfor)I ed (da))tica". Escribe un programa, que dado un mensaje encriptado, determine el mensaje original en tiempo lineal.(9° Olimpiada Mexicana de Informática).