Estructura de datos
Tema 1. Pilas

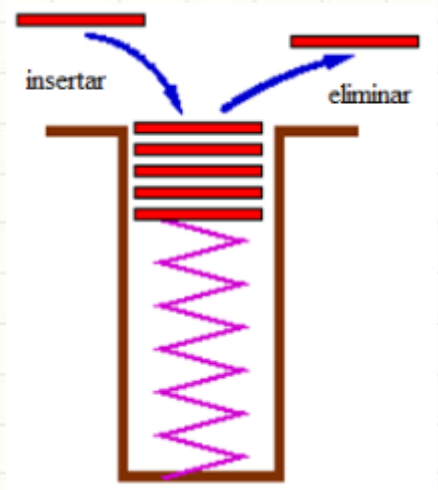
Presenta: Mtro. David Martínez Torres
Universidad Tecnológica de la Mixteca
Instituto de Computación
Oficina No. 37
dtorres@gs.utm.mx

Contenido

1. Definición y operaciones
2. Implementación estática
3. Implementación dinámica
4. Casos de estudio

1. Definición y operaciones

- Una Pila(stack) es una colección ordenada de elementos en la que se pueden insertar y eliminar por un extremo llamado **tope**.
- Por tal razón, se conoce como una estructura de datos **LIFO** (last-in, first-out)

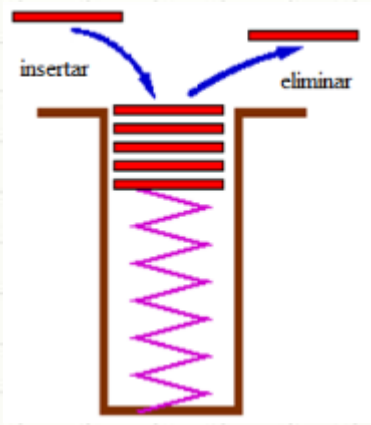


1. Definición y operaciones

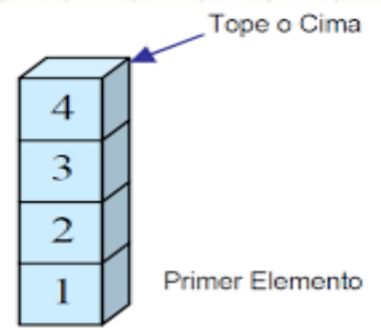
- Insertar(push) elementos a la pila
  insertar(&pila, elemento);
- Eliminar(pop) elementos de la pila
  elemento=eliminar(&pila);
- Conocer el elemento del tope de la pila
  elemento=elementoTope(pila);
- Determinar si la pila esta vacía
  bandera=vacia(pila);
- Determinar si la pila está llena
  bandera=llena(pila);

Ejemplificar:

1. insertar(pila,H);
2. insertar(pila,I);
3. insertar(pila,J);
4. eliminar(pila);
5. eliminar(pila);
6. insertar(pila,K);
7. eliminar(pila);
8. eliminar(pila);
9. insertar(pila,G);



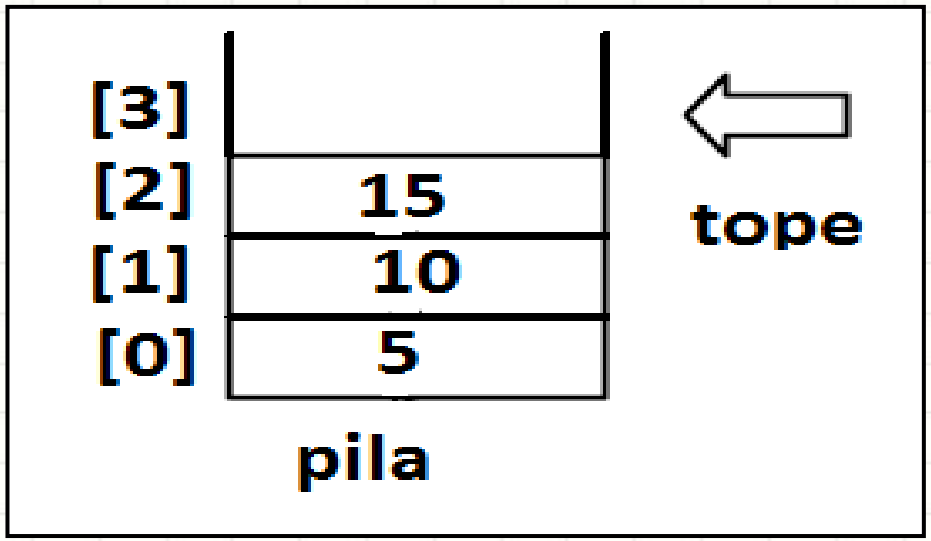
1. Definición y operaciones



- Algunas aplicaciones usando pilas:
  - Apilar libros, productos
  - Problema de torres de Hanoi
  - Leer una secuencia de caracteres desde teclado e imprimirlos al revés
  - Expresión matemática que contenga paréntesis anidados correctamente.
  - Convertir expresiones de notación infija a prefija, a postfija y viceversa.
  - Funciones recursivas
  - Sistema de estacionamiento, etc.

2. Implementación estática

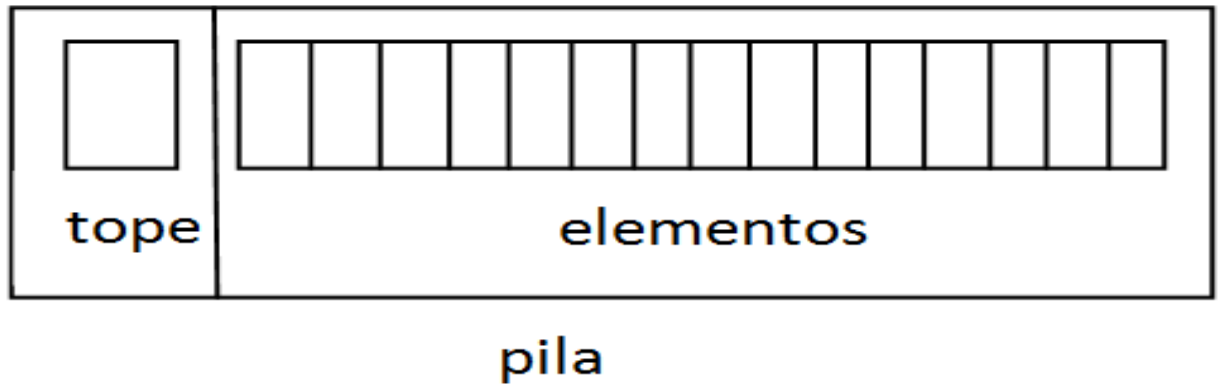
Representación estática usando arreglos.



2. Implementación estática

Otra representación estática usando arreglos.

```
#define TAM  5
typedef struct{
        int tope;
        int elementos[TAM];
}tipoPila;
```

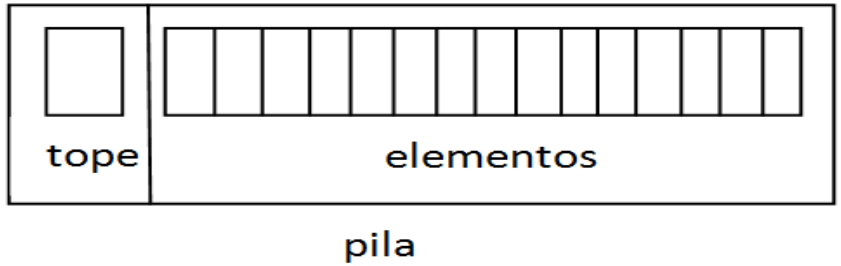


2. Implementación estática

```
#define TAM 5
typedef struct{
    int tope;
    int elementos[TAM];
}tipoPila;

void insertar (tipoPila *pilaT, int dato);
int eliminar (tipoPila * pilaT);

int main(){
tipoPila pila={-1,{0}};
insertar(&pila, 4); //push
insertar(&pila,2);
printf("El dato eliminado es: %d", eliminar(&pila));
}
```

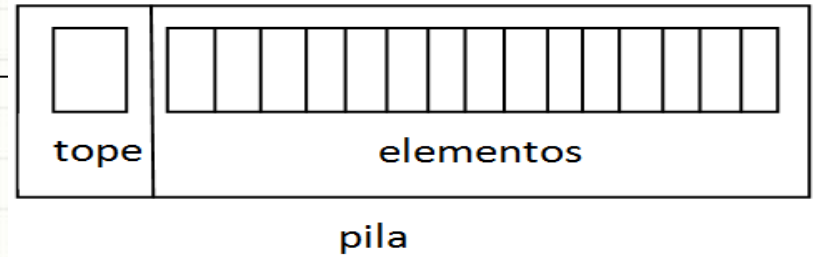


Realizar una prueba de escritorio para tener un mejor entendimiento.

Ejemplo de implementación estática

```
void insertar (tipoPila *pilaT, int dato){
//validar  invocando a función llena
pilaT->tope++;
pilaT->elementos[pilaT->tope]=dato;
}

int eliminar (tipoPila * pilaT){
int dato;
//validar invocando a función vacia
dato=pilaT->elementos[pilaT->tope];
pilaT->tope--;
return dato;
}
```

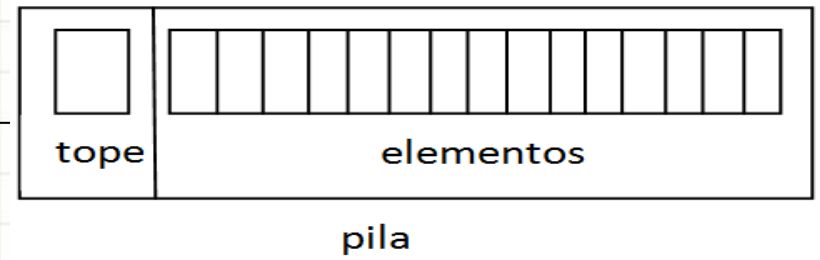


pila

Realizar una prueba de escritorio para tener un mejor entendimiento.

Ejemplo de implementación estática

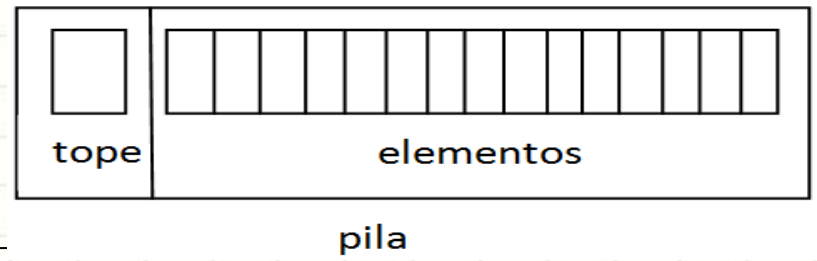
```
int llena(tipoPila pila){
int bandera=0;
if (pila.tope==TAM-1)
      bandera = 1;
return bandera
}
```



Realizar una prueba de escritorio para tener un mejor entendimiento.

Ejemplo de implementación estática

```
int vacia(tipoPila pila) {
int bandera=0;
if (pila.tope==-1)
    bandera = 1;
return 0;
}
```



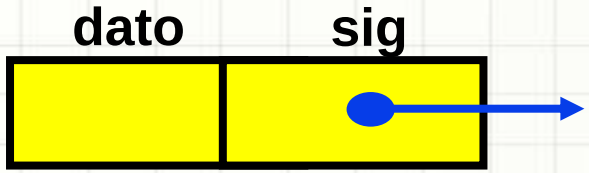
Realizar una prueba de escritorio para tener un mejor entendimiento.

Ejercicio 1. Implementación gestión de libros con pilas estáticas

- Implemente un programa que gestione libros utilizando pilas estáticas. Un libro debe contener los siguientes campos: idLibro, título, autor(es).
- En un menú de opciones debe contener las siguientes funcionalidades
  - Insertar un libro: el programa elegirá los datos de un libro de un arreglo de cadenas títulos y de un arreglo de cadenas autores de forma aleatoria; el idLibro se puede generar de forma incremental. O podrá elegir el libro de un arreglo de estructuras libros de forma aleatoria aunque se repitan libros en la pila.
  - Consultar el libro del tope
  - Eliminar el libro que desee el usuario.

- NOTA: Recuerde que todas las funcionalidades deben implementarse de acuerdo a la definición de una pila (LIFO)

3. Implementación dinámica de Pilas



Estructuras de datos que se adapten a las necesidades reales

Estructuras muy flexibles, ya sea en cuanto al orden, espacio (reservación dinámica de memoria)

Están compuestas de otras pequeñas estructuras a las que llamaremos **nodos** o **elementos**, donde cada nodo contiene:

- Los datos reales
- Uno o más punteros autorreferenciales

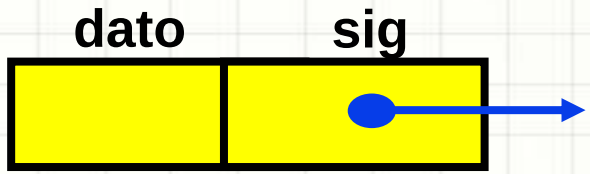
3. Implementación dinámica de Pilas

Estructura básica de un nodo para cualquier estructura dinámica sería:

```
typedef struct pila{
        int dato;
        struct pila *sig;
}nodoPila;
```

Datos reales

Puntero autorreferencial

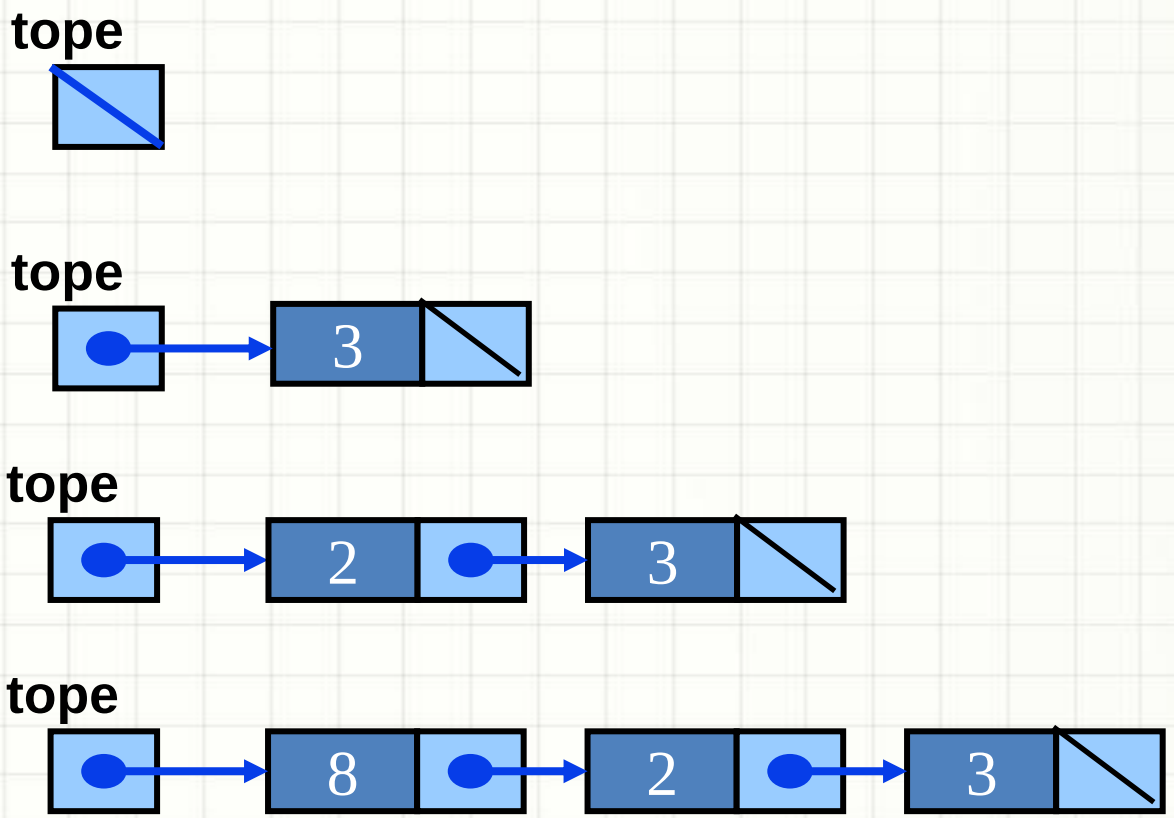


3. Implementación dinámica

```
typedef struct pila{
        int dato;
        struct pila *sig;
} nodoPila;

typedef nodoPila * nodoPilaPtr;
```

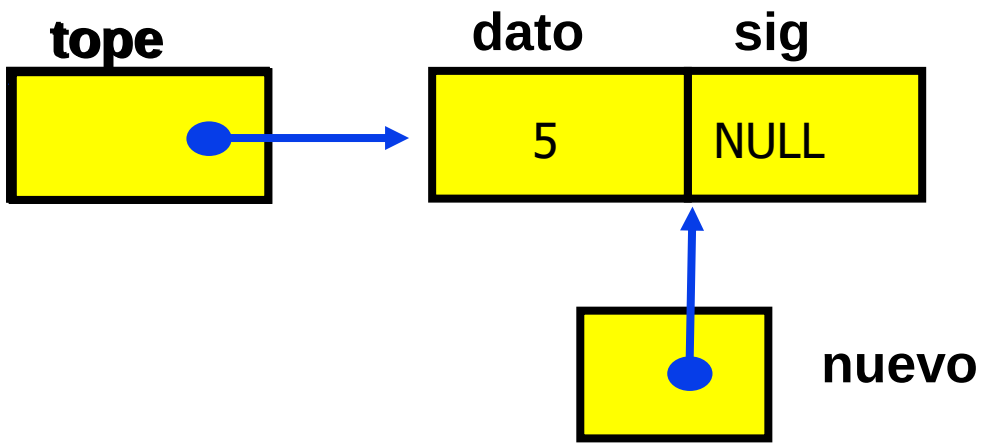
Ejemplo: Insertar a la pila vacía, el 3, 2 y 8.



3. Implementación dinámica: Insertar (push)

Algoritmo para insertar un elemento a la pila

1. Asignar espacio al **nuevo** nodo
2. Llenar los **datos reales** y asignar el contenido del apuntador **tope** a **sig**
3. Hacer que **tope** apunte a **nuevo**



3. Implementación dinámica: Insertar

Ejemplificar el siguiente código

```
typedef struct pila{
        int dato;
        struct pila *sig;
} nodoPila;
typedef nodoPila* nodoPilaPtr;

void insertar(nodoPilaPtr *tope, int
dato);
```

```
int main(){
nodoPilaPtr tope=NULL;
insertar(&tope, 3);
insertar(&tope, 2);
insertar(&tope, 8);
eliminar(&tope);
…
}
```

3. Implementación dinámica: Insertar

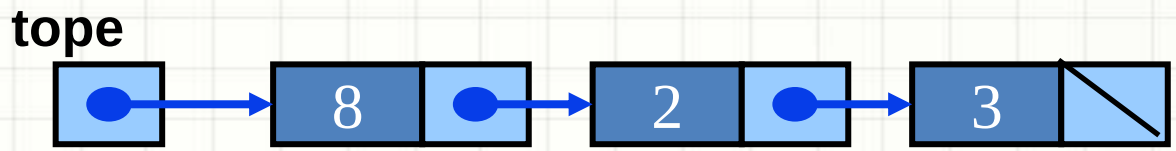
```
void insertar(nodoPilaPtr *tope, int dato){
nodoPilaPtr nuevo;

nuevo=(nodoPilaPtr) malloc(sizeof(nodoPila)); //reservar memoria

if(nuevo==NULL){
        printf("No hay memoria");
} else {
        nuevo->dato=dato;
        nuevo->sig=*tope;
        *tope=nuevo;
}
}
```

3. Implementación dinámica: eliminar (pop) un elemento

```
int eliminar(nodoPilaPtr *tope){ //validar antes de invocar
nodoPilaPtr temp;
int num;
temp=*tope;
num=temp->dato;
*tope=(*tope)->sig;
free(temp);
return num;
}
```



Sistema de un estacionamiento utilizando pilas dinámicas

Considere un tipo de dato **tipoCoche** que contenga: numPlaca(campo único), horaEntrada y horaSalida se leen de la computadora, total (calculado), con las validaciones correspondientes.
Escriba las siguientes funciones:

1. **Ingresar un coche** (ingresarCoche): La función recibe la pila, una variable de tipo **tipoCoche**. La función imprime un recibo con numPlaca, horaEntrada y precioHora.
2. **Entregar un coche** (entregarCoche): La función recibe la pila y numPlaca. Como salida imprime el reciboFinal y el usuario realiza el pago correspondiente. Considere que la función **entregarCoche** requerirá el uso de una pilaTemporal. Se puede cobrar por hora completa o por fracciones de minutos. Finalmente se escribe en un archivo el ingreso por coche (fecha,numPlaca,total)
3. **Consultar el tope de la pila** (consultaTope)
4. **Consultar ingresos** (consultarIngresos): Las consultas serán por día.
5. **Salir**: Se puede salir del sistema si existen coches en el estacionamiento, pero debe guardar los datos en un archivo binario, excepto el apuntador del nodo.

4. Casos de estudio

A continuación se presenta otro caso de estudio donde desarrollará un programa que represente el funcionamiento de convertir expresiones de notación infija a posfija.

4. Caso de estudio: Notación infija, postfija y prefija

- Aplicación que representa los diferentes tipos de pilas con sus operaciones
- De utilidad en Matemáticas, Teoría matemática de la computación, Sistemas operativos, Compiladores, etc.
- Los prefijos “pre-”, “post-” e “in-”, se refiere a la posición relativa del operador con respecto a los operandos en una expresión.
- En una conversión entre notaciones, se respeta el orden de aparición de los operandos.

4. Caso de estudio: consideraciones en notación infija

- La posición y precedencia de los operadores, así como si son asociativos por la izquierda o por la derecha
- Operadores: Suma, resta, multiplicación, división y exponenciación ($)
- Orden de precedencia:
  - exponenciación
  - multiplicación/división
  - suma/resta
- Todos son asociativos por la izquierda, excepto la exponenciación que es asociativo por la derecha

4. Caso de estudio: Notación infija

- La notación infija, también es conocida como interfija, donde en una expresión los operadores se encuentran en medio

- Ejemplos:
  - a+b
  - (a+b)*(c-d)

4. Caso de estudio: Notación prefija

- En esta notación, el operador precede a los operandos en cuestión.
- No usa paréntesis.
- Ejemplos:
  - +AB
  - -+ABC
  - *+AB-CD

4. Caso de estudio: notación postfija

- Aquí los operadores de expresiones se encuentran al final de los operandos en cuestión.
- No usa paréntesis

- Ejemplos:
  - AB+
  - ABC*+
  - AB+CD-*

4. Caso de estudio: ejercicios Infija-Postfija

INFIJA	POSTFIJA
A+B	AB+
A+B-C	AB+C-
(A+B)*(C-D)	AB+CD-*
A$B*C-D+E/F/(G+H)	AB$C*D-EF/GH+/+
((A+B)*C-(D-E))$(F+G)	AB+C*DE- -FG+$
A-B/(C*D$E)	ABCDE$*/-

4. Caso de estudio: ejercicios Infija-Prefija

INFIJA	PREFIJA
A+B	+AB
A+B-C	-+ABC
(A+B)*(C-D)	*+AB-CD
A$B*C-D+E/F/(G+H)	+-*$ABCD//EF+GH
((A+B)*C-(D-E))$(F+G)	$-*+ABC-DE+FG
A-B/(C*D$E)	-A/B*C$DE

Algoritmo para convertir una expresión infija a postfija

Ej. cad= "(10 + 3.2) – 2 * 4 $ 3"

```
opstk= la pila vacía;
while (no es fin de entrada){
        symb = siguiente carácter en la entrada
        if(symb es un operando)
                agrega symb a la cadena postfija
        else {
                while(!empty(opstk) && prcd(stacktop(opstk), symb)) {
                        topsymb=eliminar(opstk);
                        agrega topsymb a la cadena postfija;
                }//fin while
                if( empty(opstk) || symb != ')' )
                        insertar(opstk, symb);
                else //extraer el paréntesis que abre y descartarlo
                        topsymb=eliminar(opstk);
        }//fin else
}//fin while
//extracción de los operadores restantes
while(!empty(opstk)){
        topsymb=eliminar(opstk);
        agregar topsymb a la cadena postfija }
```

Reglas de precedencia de operadores

La función *prcd* acepta dos caracteres y es verdadera si el primer símbolo tiene precedencia respecto al segundo:

prcd('*','+‘) = TRUE

prcd('+','+‘) = TRUE

prcd('+','*‘) = FALSE

prcd(‘(’, op) = FALSE para cualquier operador op

prcd(op, ‘(’) = FALSE para cualquier operador op que no sea ‘)’

prcd(op, ‘)’) = TRUE para cualquier operador op que no sea ‘(’

prcd(‘)’, op) = undefined para cualquier operador op (un intento de comparar estos dos operadores indicará un error)

Conversión de infija a postfija

Realice la prueba de escritorio para

cad="**(((6-(2+3))*(3+8/2))$2)+3**"

Resultado:

6 2 3 + - 3 8 2 / + * 2 $ 3 +

Algoritmo para evaluar una expresión postfija

```
opndstk=la pila vacía
// recorre la cadena leyendo un elemento a la vez y colocando en symb
while (no se detecte el final de la entrada){
        symb=siguiente carácter de entrada;
        if(symb es un operando)
                insertar(opndstk, symb);
        else { //symb es un operador
                opnd2=eliminar(opndstk);
                opnd1=eliminar(opndstk);
                result=opnd1 symb opnd2;
                insertar(opndstk, result);
        } //fin else
}//fin while
return(eliminar(opndstk));
```

postfija= "10 3.2 + 2 4 3 $ * –"

Ejemplo de representación dinámica

```
#define INT         1
#define FLT         2
#define STRING      3

typedef union{
      int ival;
      float fval;
      char val[10];
}tipoUnion;

typedef struct{
      int tipoElem;
      tipoUnion elemento;
}tipoElemento;
```

Realice una representación gráfica

```
typedef struct pila{
      tipoElemento dato;
      struct pila *sig;
}tipoPila;

void insertarExpPila(tipoPila **, char expr[]);

int main(){
tipoPila *topeExprInf=NULL;
insertarExpPila(&topeExprInf,"( 2.2 + 2 )");
...
```

```
void insertarExpPila(tipoPila **tope, char expresion[]){
tipoElemento dato;
int i,j;
char temp[80];
i=0;
while(expresion[i]!='\0'){
     j=0;
     while(expresion[i]!=' ' && expresion[i]!='\0'){
          temp[j]=expresion[i];
          i++;
          j++;
          }
     temp[j]='\0';
     if(!isdigit(temp[0])){
          dato.tipoElem=3;
          strcpy(dato.element.val,temp);
          }
     else
          {
          j=0;
          while(temp[j]!='\0' && temp[j]!='.')
               j++;
          if(temp[j]=='\0'){
               dato.tipoElem=1;
               dato.element.ival=atoi(temp);
               }
          else {
               dato.tipoElem=2;
               dato.element.fval=atof(temp);
               }
          }
     push(tope,dato);
    if(expresion[i]!='\0')
       i++;
    }
}
```

Función que inserta una expresión en notación infija a una pila.

Referencias

1. Langsam Y., Augenstein M. J., Tenenbaum A. M. Estructuras de Datos en C. Prentice-Hall, México 1997.
2. Wirth, Niklaus. Algoritmos y estructura de Datos. Prentice-Hall, México.
3. Joyanes A. L, et. al. Estructuras de datos en C. McGraw-Hill, 2005.
4. Cairó O., Guardati S. Estructuras de datos . 3ª edición. McGraw-Hill, 2010.