Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Tema 02: TAD Pila

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom edgardoadrianfrancom

Contenido

- Descripción del TAD Pila
- Especificación del TAD Pila
  - Cabecera
  - Definición
  - Operaciones
  - Observaciones
- Implementación del TAD Pila
  - Implementación estática
  - Implementación dinámica
- Aplicaciones del TAD Pila
  - Ejemplo 01: Problema Invertir una secuencia
  - Ejemplo 02: Problema validar paréntesis correctos
- Expresiones aritméticas
  - Expresiones infijas, prefijas y postfijas
  - Evaluación de expresiones infijas
  - Conversión de infijo a postfijo
  - Evaluación de expresiones postfijas

Descripción del TAD Pila

- Uno de los conceptos mas útiles en computación es la **pila o stack.**
  - Es un conjunto de elementos, en la que:
    - Los elementos se añaden y se remueven por un solo extremo.
    - Este extremo es llamado “tope” de la pila.

- A las pilas se les llama también **listas LIFO *(last-in first-out)*** o listas “ultimo en entrar, primero en salir”.

Descripción del TAD Pila (Ejemplo)

- Cuando un empleado se va de vacaciones, le llega correo a su escritorio.
  - Las cartas se van “apilando”.
  - Al regresar de vacaciones, la ultima carta en llegar, será la primera que revisara
  - Al terminar de revisarla, la nueva carta del tope de la pila habrá cambiado
  - Del “pilo” de cartas, la mas nueva que queda, será la siguiente en ser revisada

La ultima en llegar, sera la primera en salir:
LAST IN, FIRST OUT
LIFO

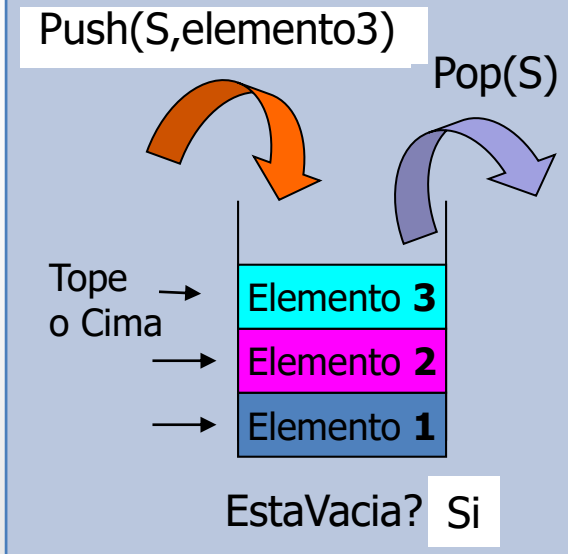
Especificación del TAD Pila

Cabecera

- ***Nombre:*** PILA *(stack)*
- ***Lista de operaciones:***
  - ***Inicializar pila (Initialize):*** Recibe una pila y la inicializa para su trabajo normal.
  - ***Empilar (Push):*** Recibe la pila y un elemento, e inserta este ultimo en el tope.
  - ***Desempilar (Pop):*** Recibe la pila, y devuelve el elemento del tope.
  - ***Es vacía (Empty):*** Recibe la pila y devuelve verdadero si esta esta vacía.
  - ***Tope pila (Top):*** Recibe la pila, y devuelve el elemento de la cima de la pila.
  - ***Tamaño pila (Size):*** Recibe la pila, y devuelve el valor del tope.
  - ***Eliminar pila (Destroy):*** Recibe una pila y la libera completamente.

Definición

- Dada una Pila llamada S



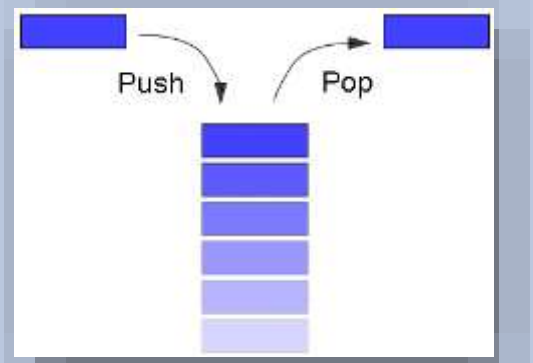
- Al empilar:
  - Cada elemento, es *"colocado"* a la pila: ***push(S,elemento).***
  - La operación ***push*** aumenta un elemento a la pila, y esta aumenta en su tamaño

- Al desempilar:
  - Se "saca" elemento a elemento de la pila: ***elemento = pop(s)***
  - La operación ***pop*** remueve el elemento del tope de la pila y lo retorna. La pila disminuye su tamaño.

ERROR: Subdesbordamiento de la pila

- **Elementos en la pila:**
  - A los elementos almacenados en la pila les corresponde una posición según fueron insertados, y esta posición se mantiene fija mientras se encuentren en la pila.

- **El TAD pila mantiene su estructura:**
  - Lo único variante es la cantidad de elementos que se encuentran “empilados” según se lleven a cabo las operaciones de empilar y desempilar. (No mutable)



Operaciones

- ***Inicializar pila (Initialize):*** ***recibe<- pila (S);***
  - *Initialize(S);*
  - Efecto: Recibe una pila y la inicializa para su trabajo normal.
- ***Empilar (Push):*** ***recibe<- pila (S); recibe<- elemento (e)***
  - *Push(S,e);*
  - Efecto: Recibe la pila y aumenta su tamaño, poniendo el elemento en la cima de la pila.
- ***Desempilar (Pop):*** ***recibe<- pila (S); retorna -> elemento***
  - *e=Pop(S);*
  - Efecto: Recibe la pila, **remueve** el elemento tope y lo **retorna.**
  - Excepción: Si la pila esta vacía, produce **error.**

- ***Es vacía (Empty):*** ***recibe<- pila (S); retorna -> boolean***
  - *b=Empty(S);*
  - Efecto: Recibe una pila y devuelve ***true*** si esta vacía y ***false*** en caso contrario.
- ***Tope pila (Top):*** ***recibe<- pila (S); retorna -> elemento***
  - *e=Top(S);*
  - Efecto: Devuelve el elemento cima de la pila.
  - Excepción: Si la pila esta vacía produce **error**
- ***Tamaño pila (Size):*** ***recibe<- pila (S); retorna -> valor del tope***
  - *n=Size(S);*
  - Efecto: Devuelve el número de elementos que contiene la pila *(Altura de la pila).*
- ***Eliminar pila (Destroy):*** ***recibe<- pila (S)***
  - *Destroy(S);*
  - Efecto: Recibe una pila y la libera completamente

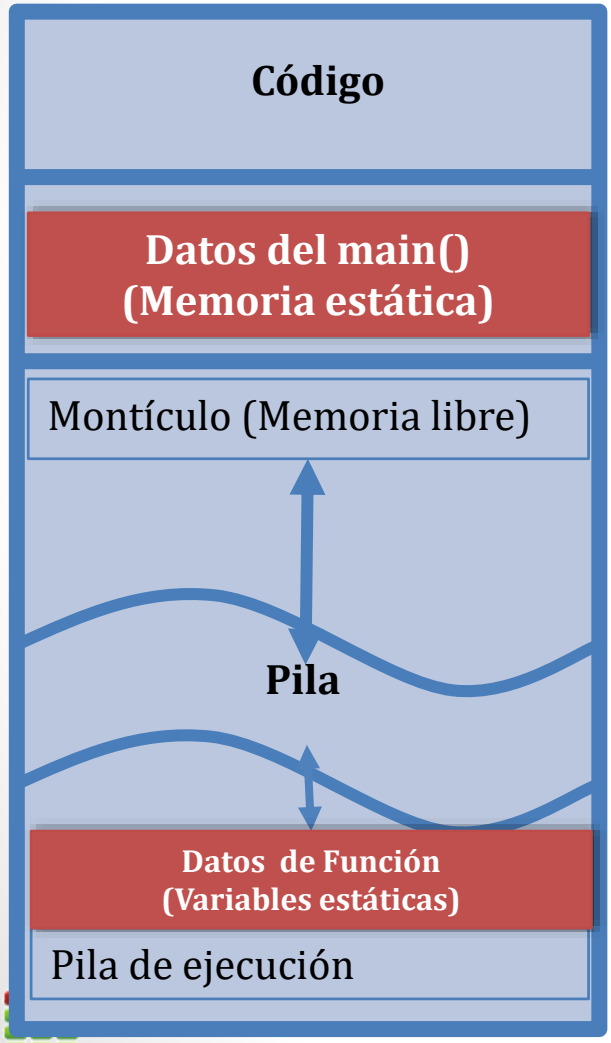
Observaciones

- El usuario de un pila debe remover hasta el ultimo elemento de una pila verificando no caer en error.
  - Una vez vacía, no se pueden “sacar o desempilar” mas elementos de la pila.
- Antes de sacar un elemento de la pila
  - Debemos saber si la pila **No esta vacía**
- El tope de la pila siempre esta cambiando
  - Deberíamos poder “revisar” el elemento **tope de la pila.**
  - Si la pila esta vacía, el tamaño de la pila es 0
- El tratar de remover elementos o acceder a elementos de una pila vacía se llama
  - **Subdesbordamiento** de la pila
- Al tratar de empilar elementos en una pila que ha llegado a su MAX_ELEMENT *(Se supone que no deberá de tener fin, aunque en la realidad no es posible)* se le llama:
  - **Desbordamiento** de la pila

Implementación del TAD Pila

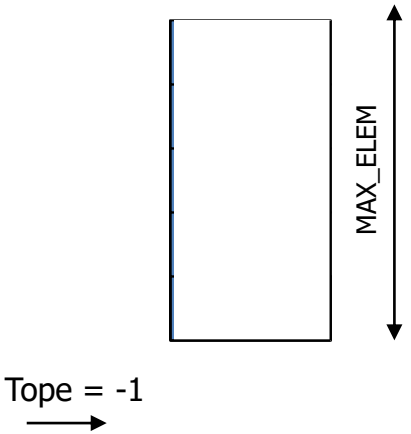
- Hay varias formas…
  - Verlo como un arreglo… con operaciones especificas
    - Un arreglo de datos el cuál se va llenando conforme se realizan inserciones y vaciando por el final del arreglo (índice Tope que lleve el control).
  - Verlo como una una Lista… empleando solo algunas de sus operaciones posibles y restringiendo el resto.
    - En la lista los nuevos nodos se pueden insertar y remover
      - Al/Del Inicio, al final, dada una posición, etc.
    - En la Pila los elementos solo se pueden **insertar al final** de la Pila
    - Y solo se pueden **remover del Final** de la Pila
- Más adelante en el curso se notará que implementaciones de la **lista simplemente enlazada** podrían usarse para implementar una pila estática o dinámica.

Implementación estática



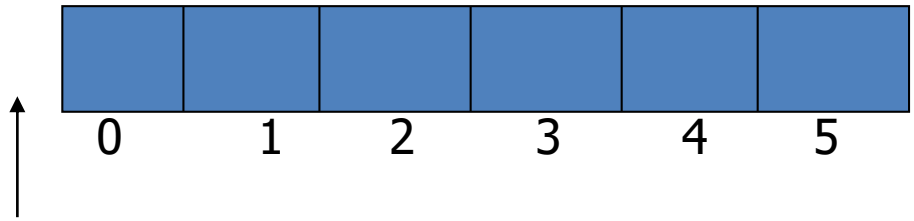
- Hablamos de una **Estructura de datos estática** cuando se le asigna una cantidad fija de memoria para esa estructura antes de la ejecución del programa.
  - La cantidad de espacio asignado para la memoria estática **se determina durante la compilación y no varía**.

Implementación estática del TAD Pila



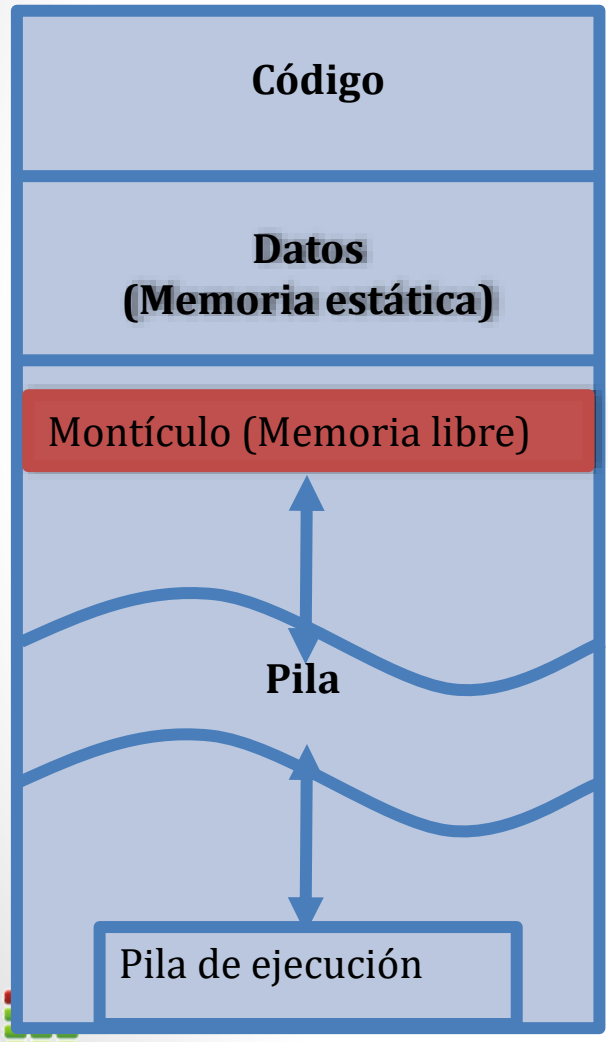
- La Pila seria un arreglo
  - typededef Array Pila;
- Y solo deberemos implementar las operaciones de
  - Push, llamando a InsertarNodoFinal
  - Pop, llamando a SacarNodoFinal
  - Top, llamando a ConsultarUltimo
  - … es decir, cada operación de la pila esconde dentro
    - Las operaciones de la Lista Contigua

Usando arreglos: Define un arreglo de una dimensión (vector) donde se almacenan los elementos.



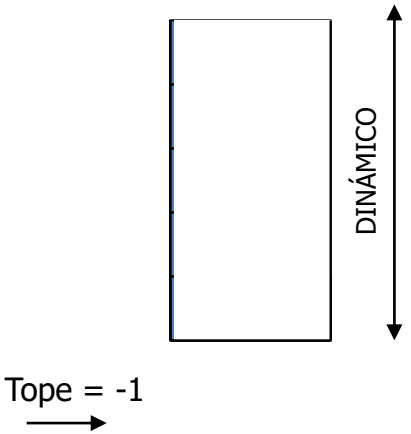
TOPE: Apunta hacia el elemento que se encuentra en el extremo de la pila. (inicialmente es -1).

Implementación dinámica



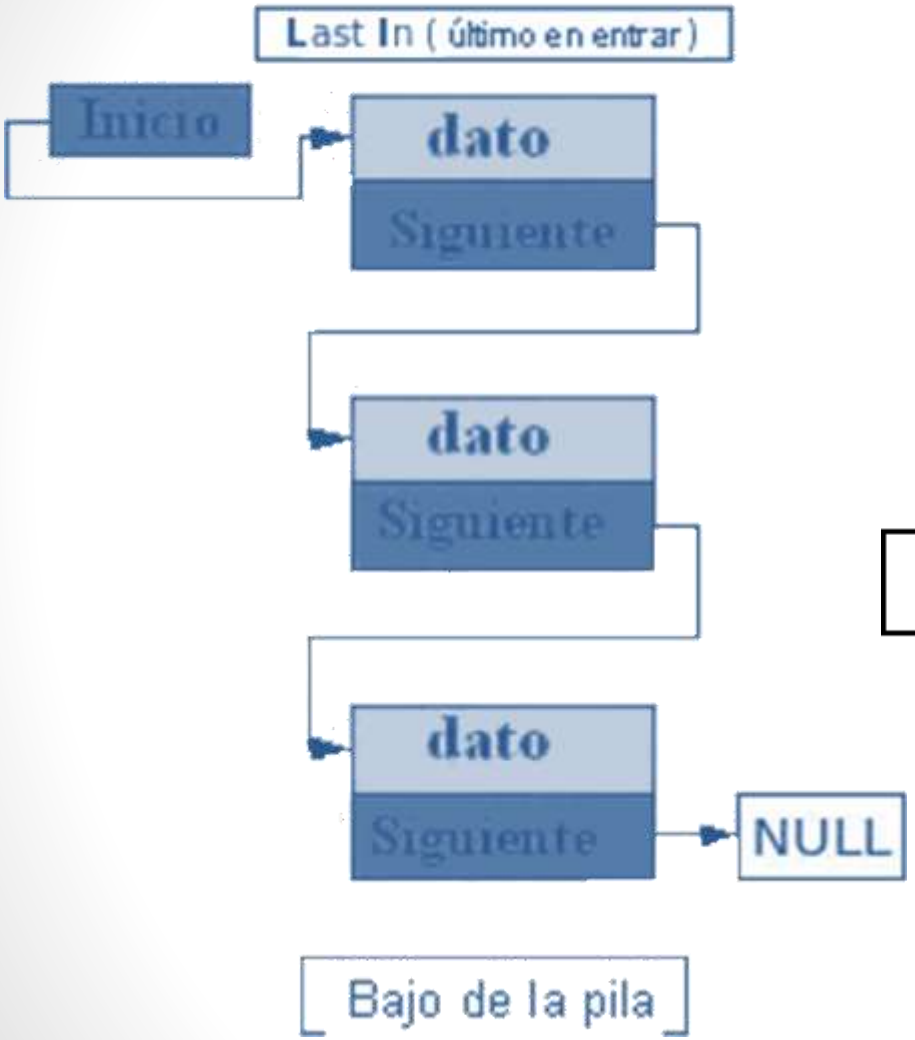
- Hablamos de una **Estructura de datos dinámica** cuando se le asigna memoria a medida que es necesitada, durante la ejecución del programa.
  - En este caso la memoria no queda fija durante la compilación.

Implementación dinámica del TAD Pila

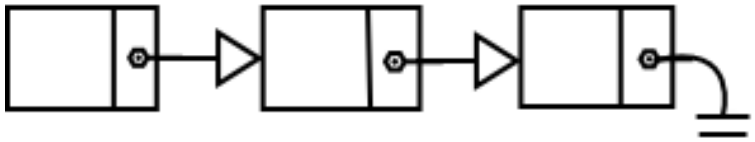


- La implementación de una pila dinámica obliga a que no sea necesario un MAX_ELEMENT, si no que se logre solicitar espacio durante la llamada a empilar para almacenar el elemento y liberar espacio durante la operación desempilar elemento.

La pila



Estructura de un nodo



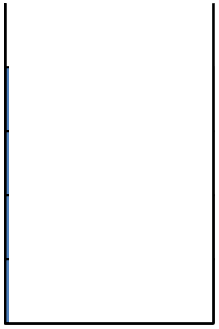
Lista enlazada

Aplicaciones del TAD Pila

- Las pilas se usan cuando para
  - Recuperar un conjunto de elementos en **orden inverso** a como se introdujeron
- Un programa debe
  - Leer una secuencia de elementos enteros
  - Luego mostrarlos en orden inverso al ingresado

- **Ejemplo 01: Problema Invertir una secuencia**
  - Si se ingresa: 1, 3, 5, 7
  - Se mostrara: 7, 5, 3, 1

Ejemplo 01: Solución



7 5 3 1

- Una vez llenada la pila,
  - Solo hay que “sacar”, elemento tras elemento
  - Hasta que la pila quede vacía

Ejemplo 02: Problema validar paréntesis correctos

- El compilador siempre sabe cuando se ha escrito un paréntesis, o una llave de mas
  - *¿Como lo hace?*

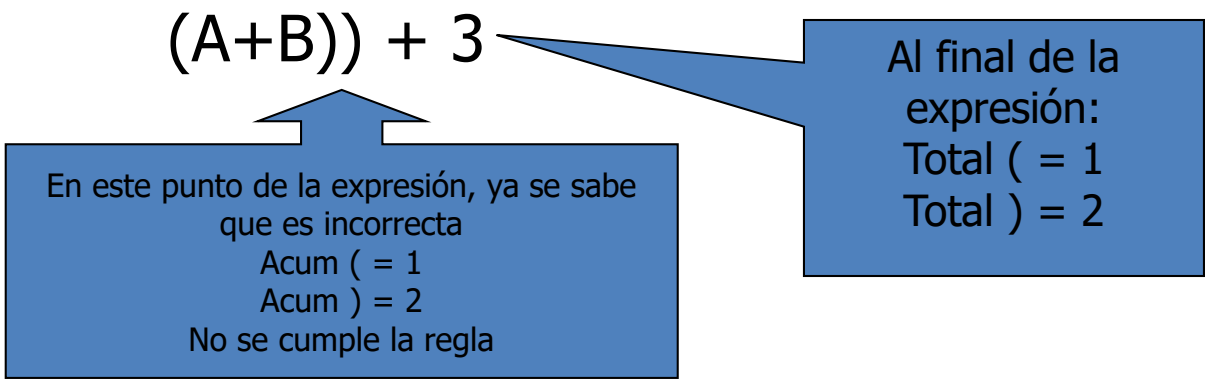
- Con el uso de pilas, expresiones escritas:
  - **(a+b))** **Mal**
  - **((a+b) * c / 4*g-h)** **OK**

- Se puede reconocer los paréntesis que no coinciden
  - *¿Como lograr esta aplicación de la pila?*

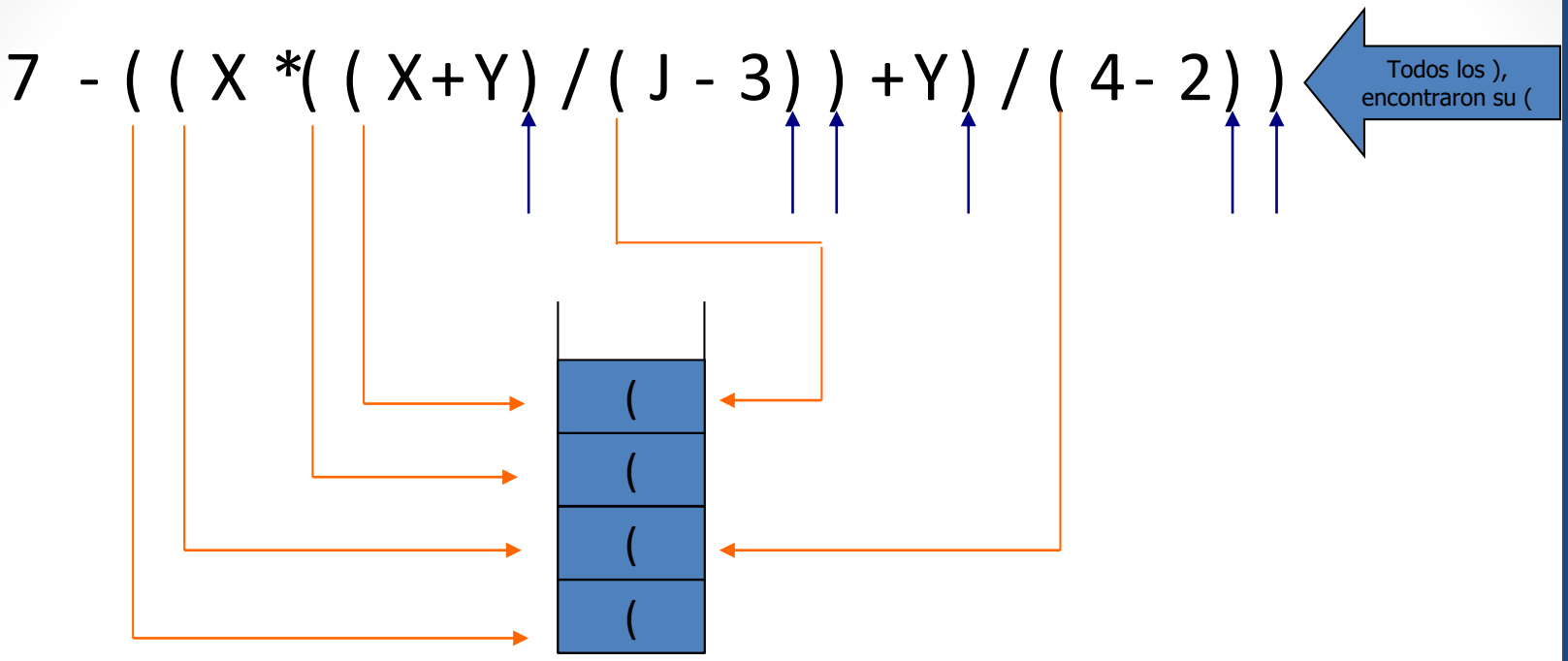
Ejemplo 02: Problema

7 - ((X* ((X+Y)/(J-3)) + Y) / (4-2.5))

- Cuando los paréntesis coinciden:
  - Al final de la expresión
    - Total paréntesis izq = Total paréntesis der y
  - En todo momento, en cualquier punto de la expresión
    - Cada paréntesis der. esta precedido de uno izq
    - Acum. paréntesis der. siempre es <= que Acum. paréntesis izq
- P.g.



Ejemplo 02: Solución



Expresiones aritméticas

Una expresión aritmética contiene constantes, variables y operaciones con distintos niveles de precedencia.

Operaciones básicas:

^ potencia

*/ multiplicación, división

+,- suma, resta

Expresiones infijas, prefijas y postfijas

Notación infija:
Los operadores aparecen en medio de los operandos.

A + B, A – 1, E/F, A * C , A ^ B , A + B + C, A+B-C

Notación prefija
El operador aparece antes de los operandos.

+ AB, - A1, /EF, *AC, ^AB, +AB+C, +AB-C

Notación postfija:
El operador aparece al final de los operandos.

AB+, A1-, EF/, AC*, AB^, AB+C+, AB+C-

Evaluación de expresiones infijas

Pasos para evaluar una expresión:

1. **Convertir a postfijo:** convertir la expresión en notación infijo a notación postfijo. (Se emplea una pila)

2. **Evaluar la expresión postfija:** *usar una pila* para mantener los resultados intermedios cuando se evalúa la expresión en notación posfijo.

Conversión de infijo a postfijo

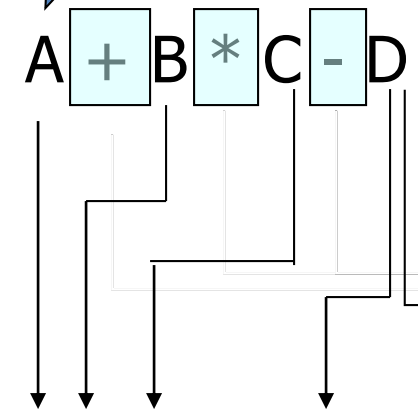
Pasos para convertir expresión infija a postfija

1. Se crea una cadena **resultado** donde se almacena la expresión en postfijo.
2. Los operandos se agregan directamente al **resultado**
3. Un paréntesis izquierdo se mete a la pila y tiene prioridad o precedencia cero (0).
4. Un paréntesis derecho saca los elementos de la pila y los agrega al **resultado** hasta sacar un paréntesis izquierdo.
5. Los operadores se insertan en la pila si:
   - La pila esta vacía.
   - El operador en el tope de la pila tiene menor precedencia.
   - Si el operador en el tope tiene mayor precedencia se saca y agrega al resultado (repetir esta operación hasta encontrar un operador con menor precedencia o la pila este vacía).
6. Cuando se termina de procesar la cadena que contiene la expresión infijo se vacía la pila pasando los elementos al **resultado**.

C infijo a postfijo

A es un operando, es añadido directamente a la nueva expresión en postfija

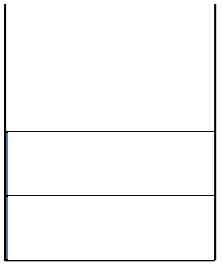
El operador de mayor precedencia es el primero en aparecer en la expresión



El operador de mayor precedencia en la expresión será el primero en aparecer en la conversión

A B C * + D -

Aquí terminamos de revisar la expresión, símbolo por símbolo.
En la pila, quedan aun operadores.
Todos se sacan y se añaden a la nueva expresión
Así termina la conversión



Conversión de infijo a postfijo

Reglas para evaluar si se incluyen paréntesis

- Para resolver este problema, podemos seguir las siguientes reglas:
  - Los paréntesis izquierdos (
    - Siempre van a ser añadidos a la pila, pase lo que pase
  - Los paréntesis derechos)
    - Significa que un ambiente de () ha sido terminado,
    - Todos los operadores de la pila, se sacan, hasta encontrar un (

Evaluación de expresiones postfijas

Reglas para evaluar una expresión postfija

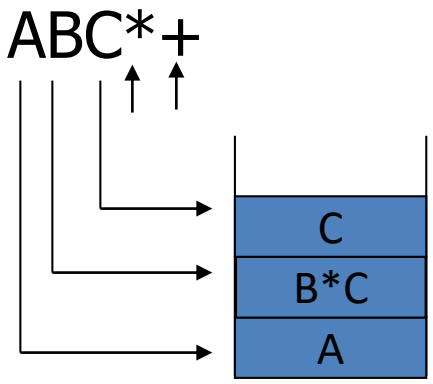
Recorrer la expresión de izquierda a derecha

1. Si es un operando
   - Almacenar el valor en la pila de valores
2. Si es un operador:
   - Obtener dos operandos de la pila de valores
   - Aplicar el operador
   - Almacenar el resultado en la pila de valores

Al finalizar el recorrido, el resultado estará en la pila de valores

Evaluación de expresiones postfijas

- Deberíamos poder “recordar” cada **operando** de la expresión y su cálculo
- Si encontramos un **operador**
  - Los dos últimos operandos recordados son los usados y “olvidados”
  - El resultado de la operación, debe ser también “recordado”
- Así, hasta que la expresión termine



A + B *C