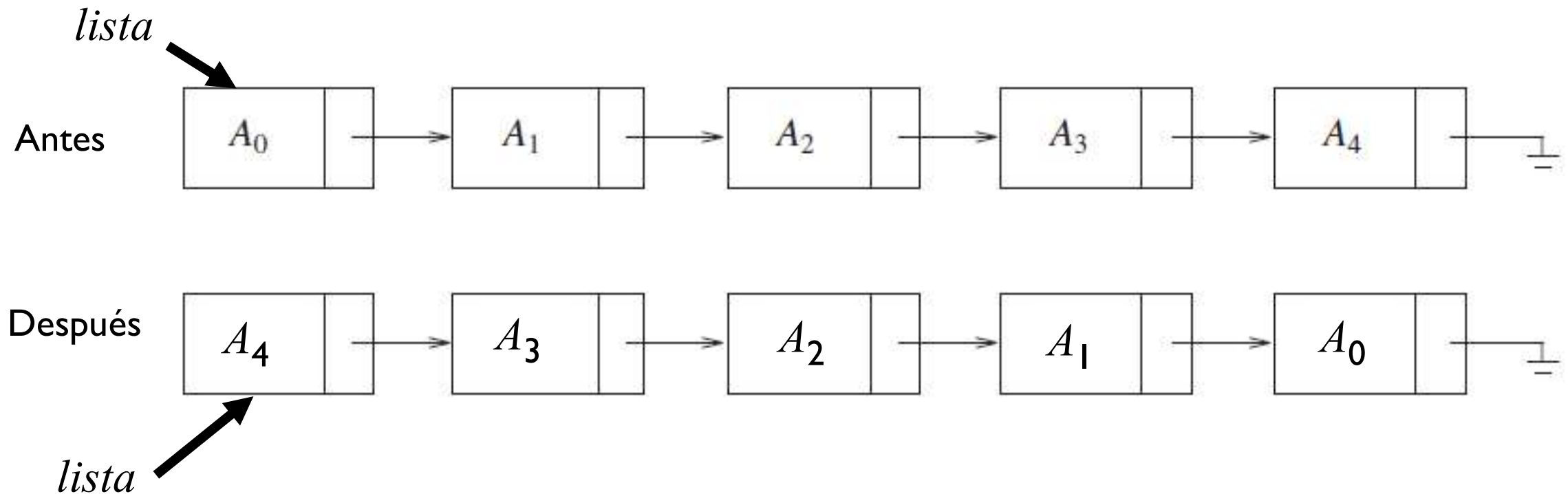


Algoritmos y Estructuras de Datos

Oscar Meza

omezahou@ucab.edu.ve

Hagamos juntos una función iterativa para invertir una lista utilizando una lista simple con apuntador al primero



Iterativo:

```
void invertir_iter(Nodo* &lista){  
    if (lista==nullptr) return;  
    Nodo* anterior;  
    Nodo* actual;  
    Nodo* tmp;  
    anterior = nullptr; actual = lista;  
    while (actual != nullptr ){  
        tmp=actual;  
        actual = actual->proximo;  
        tmp->proximo = anterior;  
        anterior = tmp;  
    }  
    lista=anterior;  
}
```

¿ Y uno recursivo que invierta la lista ?
Por supuesto, sin crear nuevos nodos

Recursivo: necesitamos de una función que agregue un **nodo** al final de una lista

```
void invertir(Nodo* &lista){
    if (lista==nullptr) return;
    Nodo* tmp = lista;
    Nodo* proximo = lista->proximo;
    tmp->proximo = nullptr;
    invertir (proximo);
    insertarCola(proximo,tmp);
    lista = proximo;
}
```

```
void insertarCola(Nodo* &lista,
                  Nodo* nodo){
    // nodo debe tener proximo en null
    if (lista == nullptr)
        lista = nodo ;
    else { // lo inserta de ultimo
        Nodo *cursor = lista;
        while (cursor->proximo != nullptr)
            cursor = cursor->proximo;
        cursor->proximo = nodo ;
    }
}
```

Hagamos juntos el siguiente ejercicio:

Tenemos dos listas simples de enteros ordenadas de menor a mayor. Las quiero combinar en una nueva lista simple ordenada con todos los elementos de las dos listas.

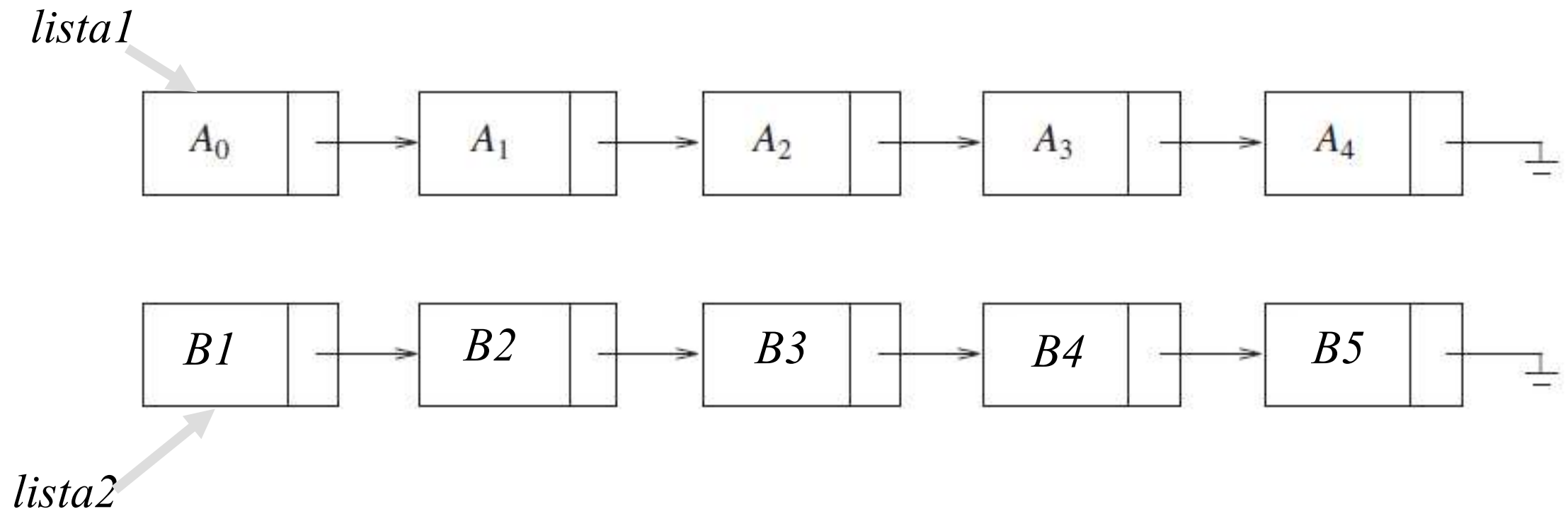
Ej:

lista1 <1, 4, 8>

lista2 <-1, 5, 10>

Resultado lista3: <-1, 1, 4, 5, 8, 10>

Tenemos dos listas simples de enteros ordenadas de menor a mayor. Las quiero mezclar en una sola lista simple ordenada con todos los elementos de las dos listas.
Sin modificar las originales.
Pensémoslo juntos....



Ej:

lista1 <1, 4, 8>

lista2 <-1, 5, 10>

Resultado lista3: <-1, 1, 4, 5, 8, 10>


```
Nodo* mezclar(Nodo* &lista1, Nodo* &lista2){ devuelve una lista mezclada
Nodo* p = nullptr; // apuntara a la lista mezclada
Nodo* cursor1 = lista1;
Nodo* cursor2 = lista2;
Nodo* ultimo = nullptr; // mantenemos el último para que sea más fácil ir
                        // insertando de último en la nueva lista
while (cursor1!=nullptr && cursor2!= nullptr)
    if (cursor1->data <= cursor2->data){
        if (p==nullptr) {// se trata diferente lista vacia
            p = new Nodo{cursor1->data,nullptr};
            ultimo = p;
        }
    else{
        ultimo->proximo = new Nodo{cursor1->data,nullptr};
        ultimo = ultimo->proximo;
    }
    cursor1=cursor1->proximo;
}
```

```
else{  
    if (p==nullptr) {  
        p = new Nodo{cursor2->data,nullptr};  
        ultimo = p;  
    }  
    else{  
        ultimo->proximo = new Nodo{cursor2->data,nullptr};  
        ultimo = ultimo->proximo;  
    }  
    cursor2=cursor2->proximo;  
}
```

```

while (cursor1!=nullptr){ // la lista 2 ya fue totalmente recorrida
                        // hay que guardar en la nueva lista los restantes de
                        // la primera

    if (p==nullptr) {
        p = new Nodo{cursor1->data,nullptr};
        ultimo = p;
    }
    else{
        ultimo->proximo = new Nodo{cursor1->data,nullptr};
        ultimo = ultimo->proximo;
    }
    cursor1=cursor1->proximo;
}

```


```

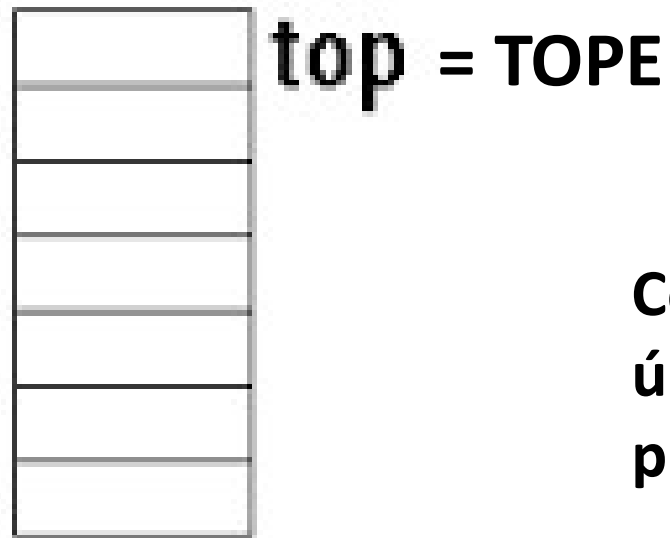
while (cursor2!=nullptr){
    if (p==nullptr) {
        p = new Nodo{cursor2->data,nullptr};
        ultimo = p;
    }
    else{
        ultimo->proximo = new Nodo{cursor2->data,nullptr};
        ultimo = ultimo->proximo;
    }
    cursor2=cursor2->proximo;
}
return p;
}

```

El tipo de dato abstracto **PILA (STACK)**

Una Pila (o Stack) es una secuencia de objetos de un mismo tipo donde sólo se tiene acceso a un solo extremo de la secuencia que llamamos tope:

EMPILAR = push  **pop = DESEMPILAR**



**Comportamiento:
último en entrar,
primero en salir**

last in first out (LIFO)

Ejemplo:

Si la pila es la secuencia $\langle 7, 3, 7, 5, 4, 2 \rangle$ supongamos que el tope es el último de la secuencia

Desempilar el tope resultaría en $\langle 7, 3, 7, 5, 4 \rangle$

Desempilar el tope de la anterior resultaría en $\langle 7, 3, 7, 5 \rangle$

Empilar 8 resultaría en $\langle 7, 3, 7, 5, 8 \rangle$

El tipo de dato abstracto **PILA (Stack en inglés)** es una secuencia de objetos del mismo tipo **Tipo** cuyas operaciones son:

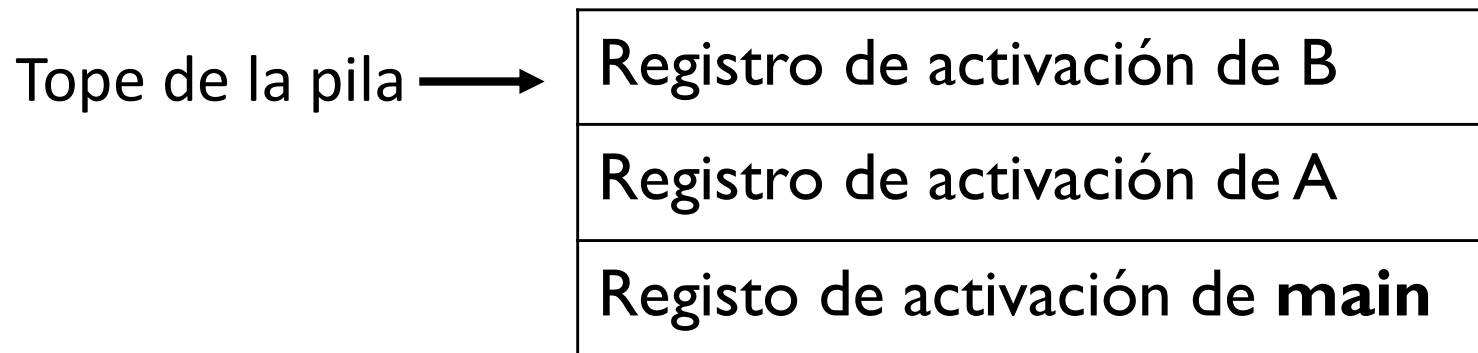
- **void empilar** (Tipo x): coloca en el tope de la pila al elemento x
- **void desempilar** (): elimina el tope de la pila
- **Tipo tope** (): devuelve el tope de la pila
- **boolean esVacia** () : devuelve verdad si la pila (secuencia) está vacía
- **Pila crearVacia** (): devuelve una pila vacía
- **int numElem** (): devuelve el número de elementos de la pila

Ejemplos de uso:

Una de las aplicaciones del tipo pila ya la habíamos visto cuando dimos recursión, los “**activation records**”. En cada llamada a una función se crea un registro de activación “activation record” con el valor de las variables locales, etc.

En sucesivas llamadas a métodos se utiliza una pila para ir guardando pista de los registros de activación:

main llama al método A y este llama a método B (se empilan los registros):



Al terminar el método B se borra ese registro y se pasa al registro de A

Otro ejemplo: Expresión aritmética en forma postfija (utilizada en Compiladores de programas).

¿Qué es una forma postfija de una expresión aritmética?

Supongamos que queremos evaluar la expresión completamente parentizada

$$((a*b)+(c / (a-b)))$$

La forma postfija de la expresión anterior es: **ab*cab-/+**

Es una secuencia de símbolos que permiten evaluar la expresión original infija utilizando una pila, como veremos.

Note que la expresión **postfija no posee paréntesis**.

¿Cómo definimos una expresión postfija a partir de una infija?

Por ejemplo de $((a*b)+(c / (a-b)))$

Hacemos una definición recursiva:

- 1) Una variable es una expresión postfija.
- 2) Vemos cual es el operador que se aplica de último en la expresión infija. En el ejemplo el +. Buscamos recursivamente la expresión postfija de la expresión a la izquierda del +, llamémosla exp1, y buscamos recursivamente la expresión postfija de la expresión a la derecha del +, llamémosla exp2.

La expresión postfija de la expresión infija original será: exp1 || exp2 || +

donde || significa concatenación de secuencias

Apliquemos la definición recursiva a $((a*b)+(c / (a-b)))$:

El operador que se aplica de último es +

La expresión postfija de $(a*b)$ es ab^*

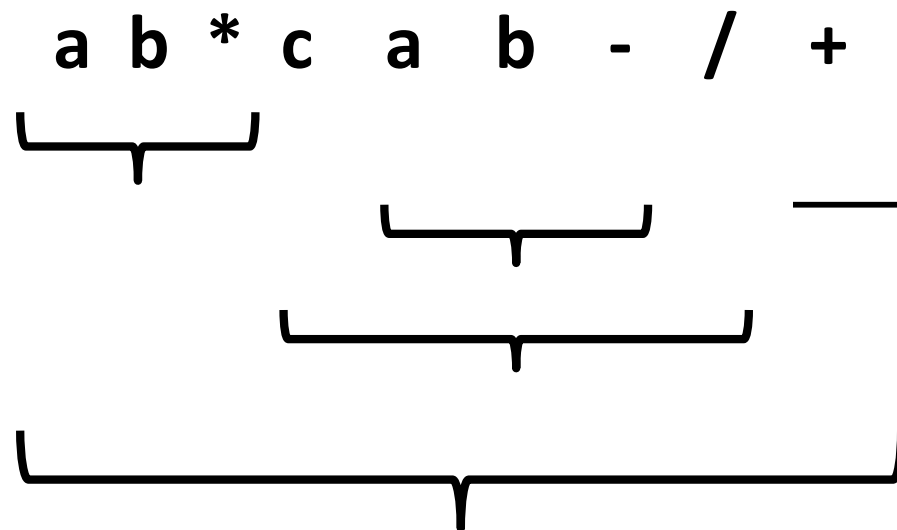
La expresión postfija de $(c / (a-b))$: c es ya la expresión postfija del operando izquierdo de $/$ y $ab-$ es la expresión postfija del operando derecho, así que la expresión postfija de $(c / (a-b))$ es $cab-/$

Por lo tanto la expresión postfija equivalente a la expresión infija original es

$$ab^*cab-/+$$

$ab*cab-/+$ es la expresión postfija equivalente a $((a*b)+(c / (a-b)))$:

Y es fácil de evaluar porque vamos leyendo de izquierda a derecha y cuando consigamos un operador se lo aplicamos a los dos operandos precedentes



Al conseguir $-$ hace la resta de **a** menos **b** y lo convierte en un operando

Hallemos juntos la expresión postfija de ($(a+(b+(c*d))) + (a*(d+c))$)

$$((a+(b+(c*d))) + (a*(d+c)))$$

la expresión postfija de $((a+(b+(c*d))) + (a*(d+c)))$ es **$abcd*++adc+*+$**

Hallemos juntos la expresión postfija de $((6*(8/2)) * 10)$ y luego evaluemos la expresión postfija

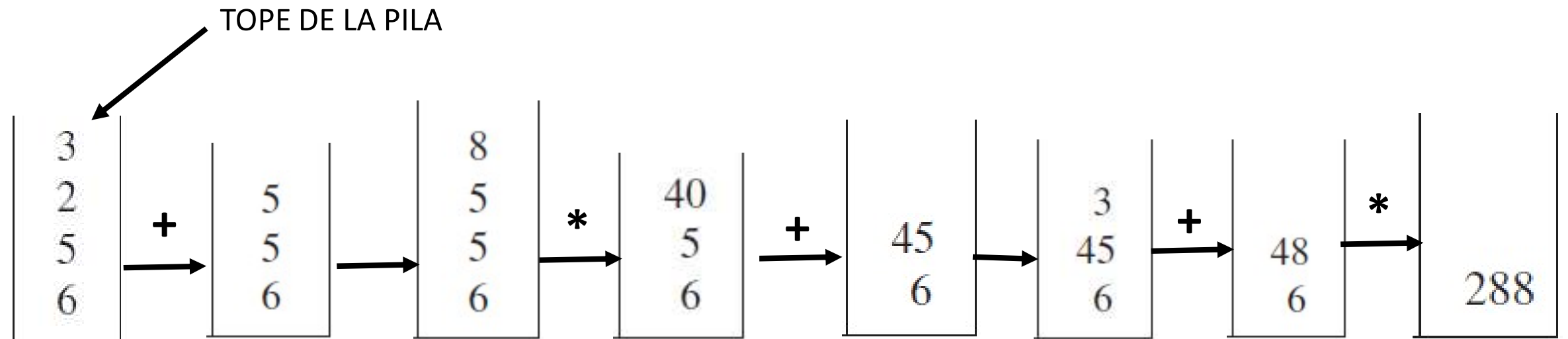
$$((6*(8/2)) * 10)$$

¿ Cómo evaluar la expresión postfija utilizando una pila 6 5 2 3 + 8 * + 3 + * ?

(suponemos que cada número es un operando aparte)

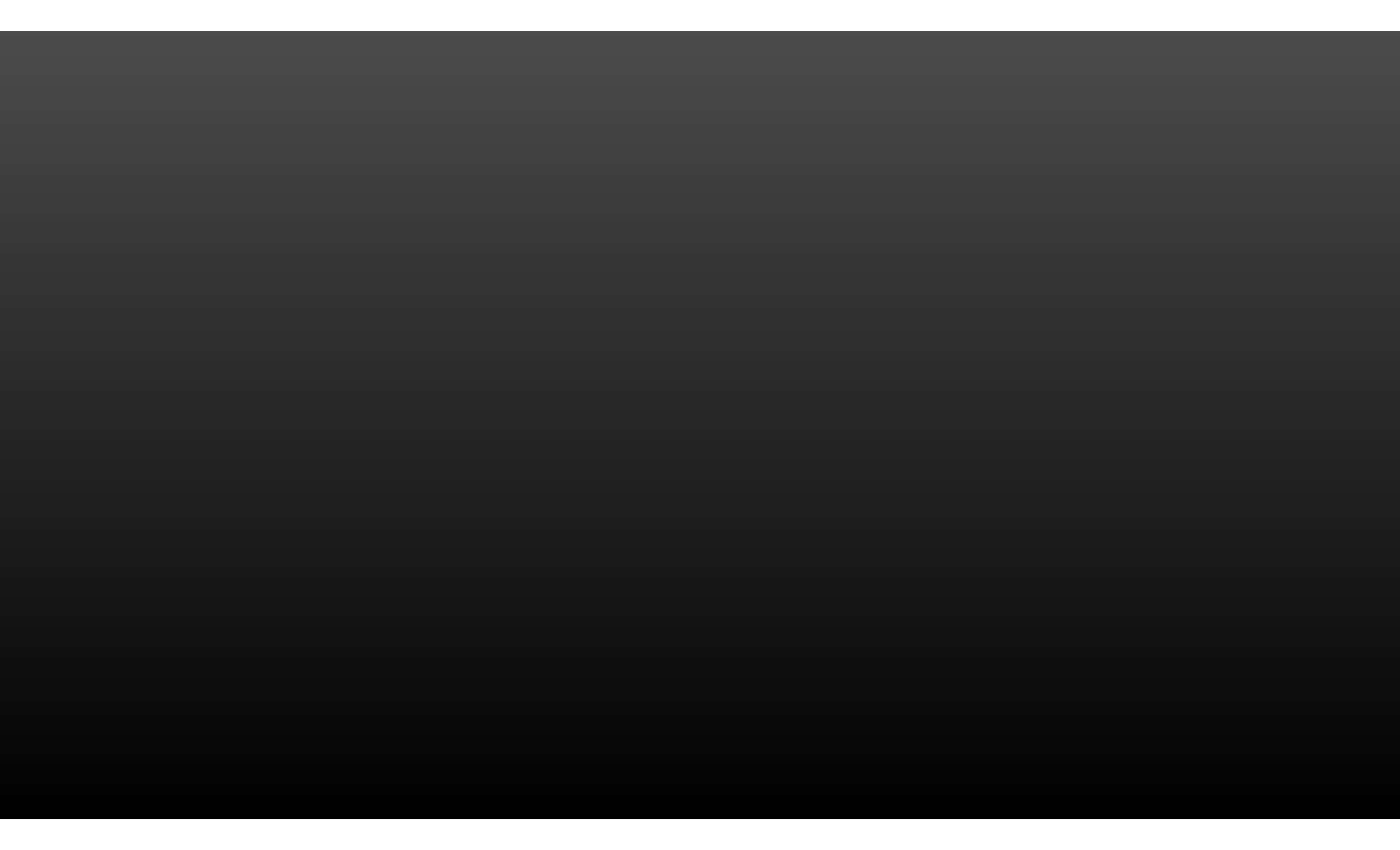
Vamos leyendo los caracteres de izquierda a derecha y los vamos guardando en una pila hasta llegar a un operador y aplicamos el operador a los dos elementos en el tope y guardamos el resultado en la pila.

Comenzamos con que ya hemos leído 6, 5, 2 y 3



Hagamos juntos utilizando una pila:

evaluar la expresión postfija: $6\ 5+ 2\ 3 + 8\ * \ * 3 +$



evaluar la expresión postfija: $6\ 5+ 2\ 3 + 8 * * 3 +$

¿Cuánto da...? 443

Otras aplicaciones y algoritmos sobre pilas:

- Convertir una expresión infija completamente parentizada, en una postfija usando el tipo pila
- Mover recursivamente una pila a otra y que queden sus elementos en el mismo orden
- Mas adelante veremos más usos del tipo Pila, cuando veamos Arboles y Grafos

Convertir una expresión infija completamente parentizada, en una postfija.

Ejemplo: la expresión $((A*B)/((C-D)*E))$:

Ir tratando elemento a elemento de izquierda a derecha, comenzamos con una pila P y una lista L vacías. En L quedará la expresión postfija equivalente

Mientras existan elementos a considerar:

- Si leemos paréntesis izquierdo lo empilamos en P
- Si leemos operando lo colocamos de último en la lista L
- Si leemos operador, lo empilamos en P
- Si leemos paréntesis derecho tomamos el operador en el tope de P y lo colocamos de último en la lista L, lo desempilamos, y luego desempilamos el paréntesis izquierdo

Corramos juntos el algoritmo con la expresión $((A*B)/((C-D)+E))$:

Corramos juntos el algoritmo con la expresión $((A*B)/((C-D)+E))$:

PREGUNTAS???

