ESTRUCTURA DE DATOS

APLICACIONES DE LAS PILAS

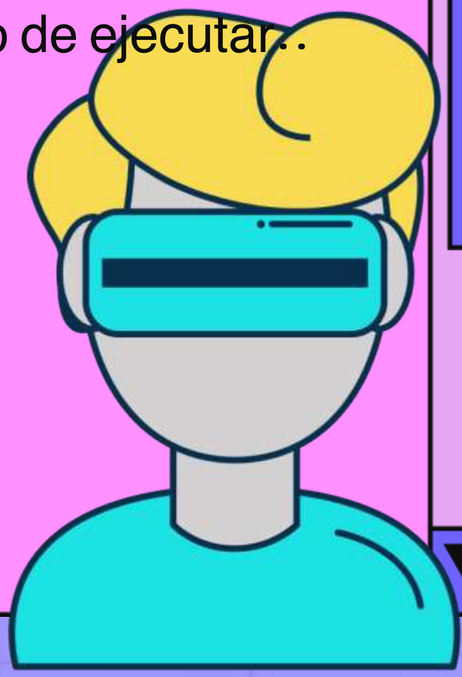
FLORES DOMINGUEZ ANGEL GABRIEL
TEZOCO CRUZ PEDRO

LLAMADAS A SUBPROGRAMAS

LLAMADAS DE SUBPROGRAMAS

Esta operación se consigue disponiendo las direcciones de retorno de una pila

Cuando dentro de un programa se realizan llamadas a subprogramas, el programa principal debe recordar el lugar donde se hizo la llamada, de modo que pueda retornar allí cuando el subprograma se haya terminado de ejecutar.



RECURSIVIDAD

¿QUE ES LA RECURSIVIDAD?

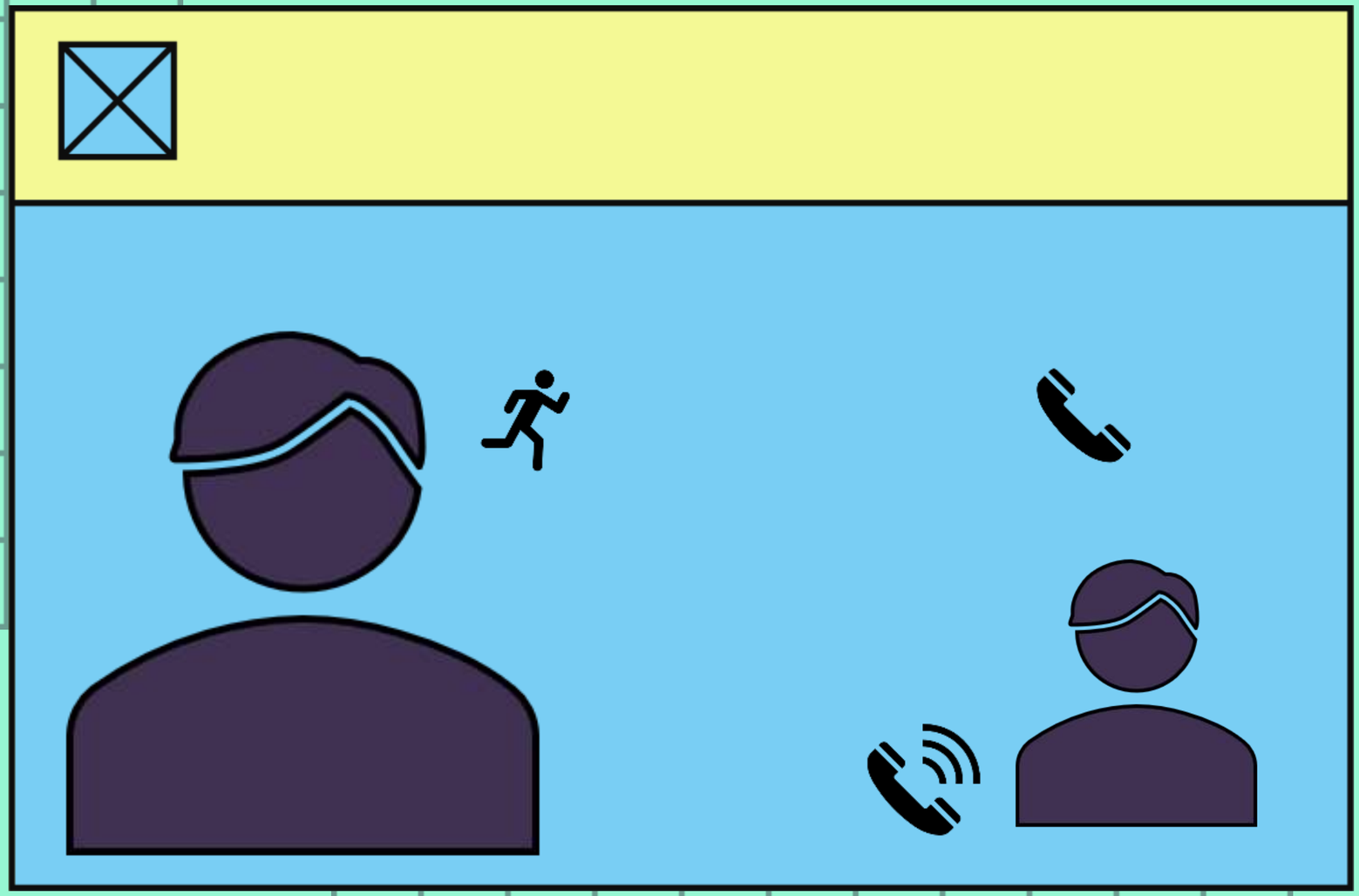
En general, la recursividad es el proceso de definir algo en términos de sí mismo y es algo similar a una definición circular.

El componente clave de un método recursivo es una declaración que ejecuta una llamada a sí mismo. La recursividad es un poderoso mecanismo de control.

CONCEPTOS CLAVE

- no hace una nueva copia del método.
- podrían causar un desbordamiento de la pila.
- algunos tipos de algoritmos se pueden implementar de forma más clara y más recursiva de lo que pueden ser iterativamente.

¿COMO FUNCIONA?



New Tab

TORRE DE HANOI

```java
public static void hanoi(int discos, int inicio, int apoyo, int destino) {
    if (discos == 1) {
        System.out.println("Mueve el disco de la torre " + inicio + " a la torre " + destino);
    } else {
        hanoi(discos - 1, inicio, destino, apoyo);
        System.out.println("Mueve el disco de la torre " + inicio + " a la torre " + destino);
        hanoi(discos - 1, apoyo, inicio, destino);
    }
}
```

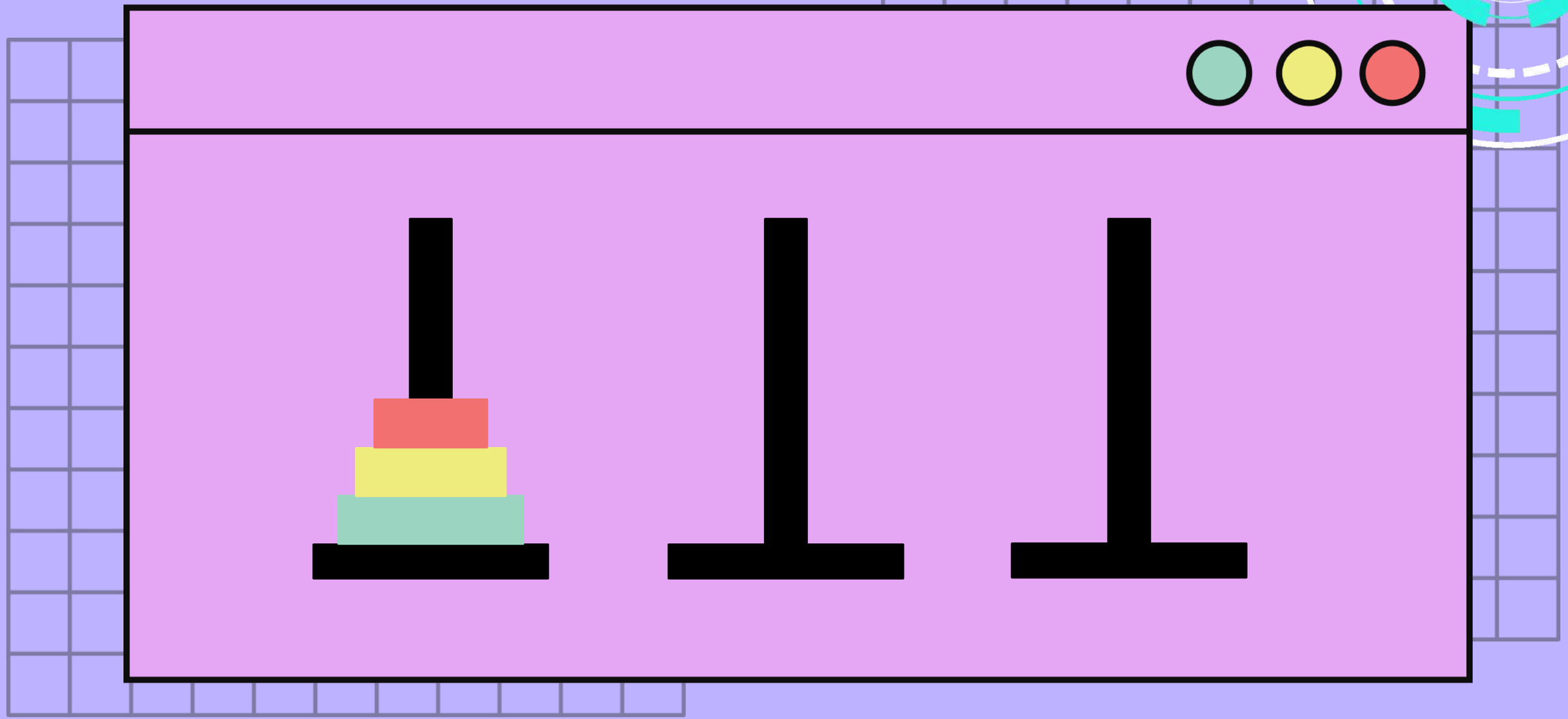
```
hanoi(3,1,2,3);
hanoi(discos: 3,inicio: 1,apoyo: 2,destino: 3) {
    if (discos 3 == 1) {
    ...
    } else {
        hanoi(discos - 1: 3-1: 2, inicio:1, destino: 3, apoyo: 2);
            hanoi(discos: 2,inicio: 1,apoyo: 3,destino: 2){
                if (discos 2 == 1) {
                ...
             } else {
                hanoi(discos - 1: 2-1: 1, inicio:1, destino: 2, apoyo: 3);
                    hanoi(discos: 1,inicio: 1,apoyo: 2,destino: 3){
                    if (discos 1 == 1) {
                            System.out.println("Mueve el disco de la torre " + inicio: 1 + " a la torre " + destino: 3);
                        } else {
                          ...
                        }
                    }
            System.out.println("Mueve el disco de la torre " + inicio:1 + " a la torre " + destino: 2);
            hanoi(discos - 1: 2-1: 1, apoyo: 3, inicio: 1, destino: 2);
                    hanoi(discos: 1,inicio: 3,apoyo: 1,destino: 2){
                    if (discos 1 == 1) {
                            System.out.println("Mueve el disco de la torre " + inicio: 3 + " a la torre " + destino: 2);
                        } else {
                          ...
                        }
                    }
            }
```

```
    System.out.println("Mueve el disco de la torre " + inicio: 1 + " a la torre " + destino: 3);
    hanoi(discos - 1: 3-1: 2, apoyo:2, inicio: 1, destino: 3);
        hanoi(discos: 2,inicio: 2,apoyo: 1,destino: 3){
                if (discos 2 == 1) {
                    ...
                } else {
                    hanoi(discos - 1: 2-1: 1, inicio: 2, destino: 3, apoyo: 1);
                    hanoi(discos: 1,inicio: 2,apoyo: 3,destino: 1){
                    if (discos 1 == 1) {
                            System.out.println("Mueve el disco de la torre " + inicio: 2 + " a la torre " + destino: 1);
                    } else {
                        ...
                    }
                }
        System.out.println("Mueve el disco de la torre " + inicio: 2 + " a la torre " + destino: 3);
        hanoi(discos - 1: 2-1: 1, apoyo:1, inicio: 2, destino: 3);
                    hanoi(discos: 1,inicio: 1,apoyo: 2,destino: 3){
                    if (discos 1 == 1) {
                        System.out.println("Mueve el disco de la torre " + inicio: 1 + " a la torre " + destino: 3;
                    } else {
                        ...
                    }
                }
        }
    }
}
```

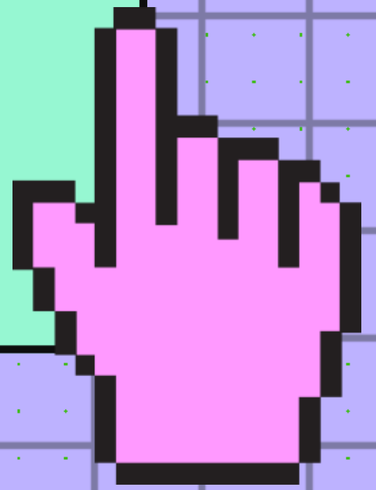
New Tab

EJECICION DE TORRE DE HANOI

```
run:
Mueve el disco de la torre 1 a la torre 3
Mueve el disco de la torre 1 a la torre 2
Mueve el disco de la torre 3 a la torre 2
Mueve el disco de la torre 1 a la torre 3
Mueve el disco de la torre 2 a la torre 1
Mueve el disco de la torre 2 a la torre 3
Mueve el disco de la torre 1 a la torre 3
BUILD SUCCESSFUL (total time: 2 seconds)
```



EQUILIBRADO DE SIMBOLOS



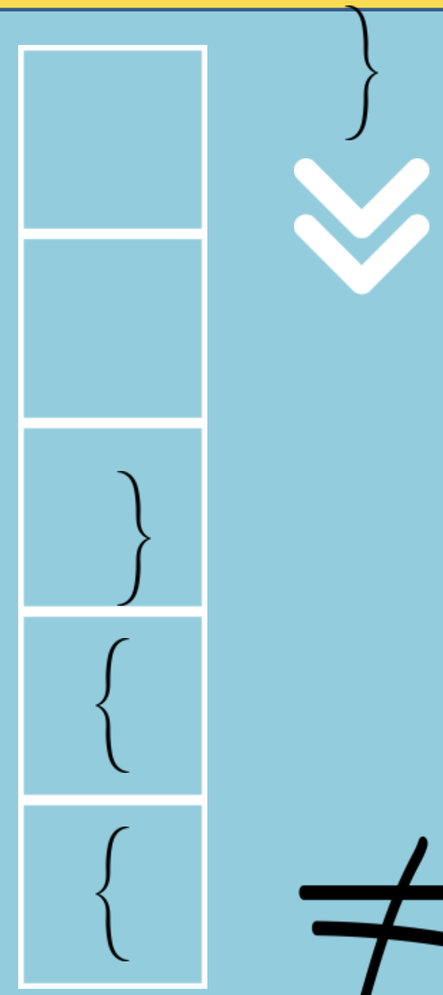
¿QUE ES EL EQUILIBRADO DE SIMBOLOS?

Utilizado por los compiladores para comprobar la sintaxis: por cada elemento clave de apertura tiene que haber uno de cierre.

Ejemplo: (), [], {}, begin-end. o En una pila se guardan los elementos (palabras o símbolos clave). Si el elemento leído se cancela con el del tope de la pila se elimina (pop()). Si no, se mete en la pila (push()). o Si al finalizar el análisis la pila está vacía, esa sintaxis es correcta.

¿COMO FUNCIONA?

- Crear una pila vacía.
- Leer la expresión de izquierda a derecha.
- Si el símbolo es una apertura de paréntesis, corchete o llave, agregarlo a la pila.
- Si el símbolo es un cierre de paréntesis, corchete o llave, verificar si la pila está vacía o si el último símbolo en la pila coincide con el tipo de cierre. Si no es así, la expresión no está equilibrada. Si coincide, sacar el último símbolo de la pila.
- Al finalizar la lectura de la expresión, verificar si la pila está vacía. Si no lo está, la expresión no está equilibrada.

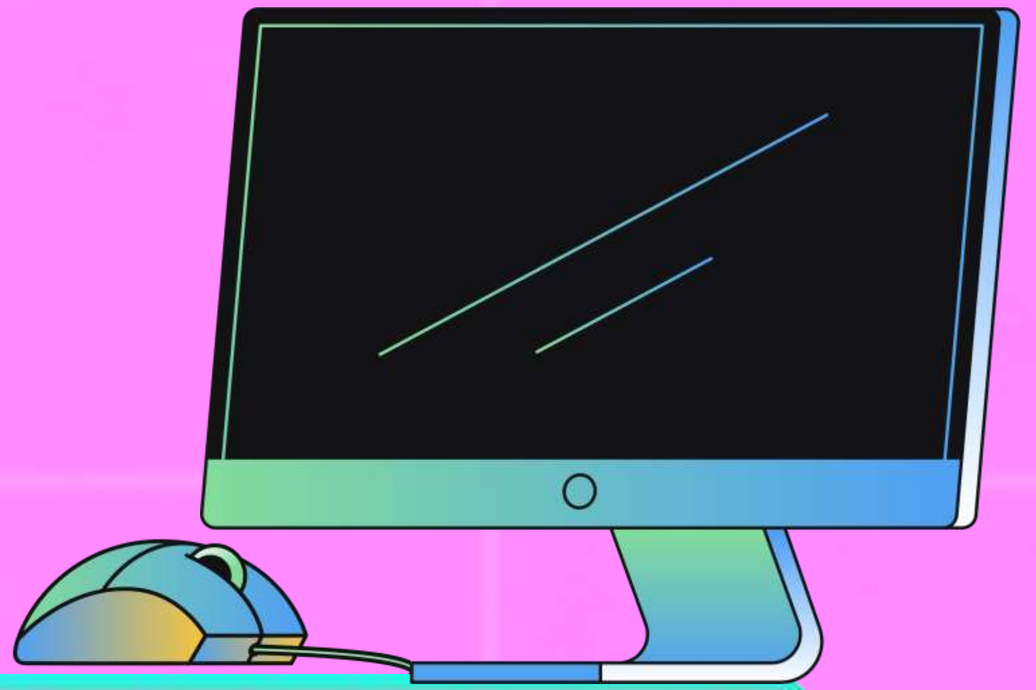



Untitled - TextEdit
File Edit View Help
TRATAMIENTOS DE EXPRESIONES ARITMETICAS

EXPRESIONES ARITMETICAS

Una expresión aritmética está formada por operandos y operadores, como Por ejemplo (a + b) - c * d. En este caso +, - * son los operadores y a, b, c, d los operandos. Esta forma de escribir una expresión, se denomina notación usual o infija

Existen otras formas de escribir expresiones aritméticas, en el que se diferencian por la situación de los operadores respecto de los operandos.



¿COMO FUNCIONA?

Untitled - TextEdit

File Edit View Help

PERFILES

(ABC) *-+/ (ABC)

POR SU ATENCION...

¡GRACIAS!