Un lenguaje de programación orientado a pila es un lenguaje que usa un modelo de máquina de pila para pasar los parámetros. Varios lenguajes de programación entran en esta descripción, notablemente Forth, RPL y PostScript, y también muchos lenguajes ensamblador (pero a un nivel muy inferior).

Los lenguajes de programación orientados a pila operan sobre uno o más pilas que pueden responder a diferentes propósitos. Debido a esto, las construcciones en otros lenguajes de programación pueden necesitar ser modificadas para el uso en un lenguaje de programación orientado a pila. Sumado a esto, algunos lenguajes de programación orientados a pilas operan en notación polaca inversa (RPN) o *notación de postfijo* – es decir, los argumentos o parámetros para un cierto comando se indican antes del comando real en sí mismo. Por ejemplo, para multiplicar 2 por 3, en notación polaca inversa uno diría «2, 3, multiplica» en vez de «multiplica, 2, 3» (notación de prefijo o notación polaca) ó «2 multiplica 3» (notación de infijo).

Asuma que tenemos un lenguaje de programación de posfijo basado en pila. El PostScript es uno de tales lenguajes. Para entender cómo trabaja un lenguaje orientado a pila, en el cálculo de una expresión tal como `2 3 mul`, podemos utilizar un experimento de pensamiento simple.

Digamos que usted está colocando en el extremo de una banda transportadora (la entrada), sobre la cual alguien ha colocado (en secuencia) placas marcadas 2, 3, y `mul`. Usted puede tomar la placa en el extremo del transportador (2), que es la primera, pero no puede ver o tomar otras placas del transportador hasta que haga algo con la placa que acaba de tomar. La única manera que puede almacenar las placas es en un pila, como la pila de platos en la cocina, y solo puede agregar o quitar una placa encima de la pila, no en el centro. También tiene una fuente de placas en blanco (y un marcador), y puede desechar las placas (pero esto es permanente). ¿Puede usted realizar el cálculo?

Sí. Usted toma la placa 2 y la pone en la pila, después toma la placa 3 y la pone en la pila. Después, usted toma la placa del `mul`. Esto es una instrucción para usted. Usted tomará las dos placas superiores de la pila, multiplicará sus etiquetas (2 y 3), y escribirá el resultado (6)

en una nueva placa. Las dos placas viejas (2 y 3) y la placa del `mul` entonces son descartadas, y la nueva placa con la respuesta es puesta en la pila. Sin más placas que queden en el transportador, el resultado del cálculo (6) es mostrado en la placa en el tope de la pila.

Esto es por supuesto un cálculo muy simple. ¿Qué pasaría si quisiéramos calcular algo como `(2 + 3) x 11 + 1`? Si primero lo escribimos en forma de posfijo, es decir, `2 3 add 11 mul 1 add`, podemos realizar el cálculo en exactamente la misma manera y alcanzar el resultado correcto. Los pasos del cálculo son mostrados en la tabla abajo. Cada columna muestra un elemento de la entrada (la placa en el extremo del transportador), y el contenido de la pila después de procesar esa entrada.

Input	2	3	add	11	mul	1	add
Pila	2	3 2	5	11 5	55	1 55	56

Después de procesar toda la entrada, vemos que la pila contiene 56, que es la respuesta.

De esto, podemos concluir lo siguiente: un lenguaje de programación basado en pila tiene solamente una forma de manejar datos, tomando una sola pieza de datos del tope de la pila, procedimiento conocido como *pop*, y poniendo datos en el tope de la pila, procedimiento conocido como *push*. Cualquier expresión que pueda ser escrita «convencionalmente» o en otro lenguaje de programación puede ser escrita en forma de posfijo (o prefijo) y así ser favorable de ser interpretada por un lenguaje de programación orientado a pila.

Manipulación de la pila

Puesto que la pila es la forma de la manipulación de los datos en un lenguaje de programación orientado a pila, frecuentemente estos lenguajes proporcionan una cierta clase de operadores de manipulación de la pila. Comúnmente proveen `dup`, para duplicar el elemento en el tope de la pila, `exch` (o `swap`), para intercambiar elementos en el tope de la pila (el primero se convierte en el segundo y el segundo se convierte en el primero), `roll`,

para cíclicamente permutar elementos en la pila o en parte de la pila, `pop` (o `drop`), para desechar el elemento en el tope de la pila, `push` para agregar un elemento en el tope de la pila, y otros. Todos estos llegan a ser la clave para estudiar los procedimientos.

Diagramas de efecto de la pila

Como ayuda para entender el efecto de una sentencia, es usado un corto comentario mostrando el tope de la pila antes y después de la sentencia. el tope de la pila está a la derecha si hay múltiples elementos (la pila crece de izquierda a derecha). Esta notación es comúnmente usada en el lenguaje Forth, donde los comentarios están encerrados entre paréntesis.

```
( antes -- después )
```

Por ejemplo, las operaciones de pila básicas del Forth son descritas así:

```
dup  ( a -- a a )
drop ( a -- )
swap ( a b -- b a )
over ( a b -- a b a )
rot  ( a b c -- b c a )
```

Y la función `fib` se describe abajo:

```
fib  ( n -- fib )
```

Otro ejemplo mas práctico (En una hp 32s):

```
 Resolución de una ecuación de segundo grado
```

```
LBL A
IMPUT A
IMPUT B
IMPUT C
RCL B
X²
-4
RCL A
X
RCL C
X
+
SQRT
STO Z
IMPUT Z
LBL Y
```

```
    RCL Z
    RCL B
    +/-
    +
    2
    RCL A
    X
    ÷
    STOP
    DSE 0
    GTO Z
    RTN
    LBL Z
    RCL Z
    +/-
    STO Z
    GTO Y
    RTN
La variable "o" es 2.000
```

Análisis del modelo de lenguaje

El simple modelo proporcionado en un lenguaje de programación orientado a pila permite a las expresiones y a los programas ser interpretados simplemente y ser teóricamente evaluados mucho más rápidamente, puesto que no se necesita hacer ningún análisis de sintaxis, solamente un análisis léxico. La manera en que son escritos los programas tiende bien por sí mismo a que sean interpretados por máquinas, esta es la razón por la que el PostScript se adapta bien para su uso en las impresoras. Sin embargo, la manera levemente artificial de la escritura de los programas PostScript puede dar lugar a una barrera inicial para

entender a éste y otros lenguajes de programación orientados a pilas.

Mientras que la capacidad de sombrear/ocultar las definiciones incorporadas y otras al sobreescribirlas (overriding), puede hacer las cosas difíciles de depurar – y el uso irresponsable de esta característica puede resultar en un comportamiento imprevisible – esto puede hacer también cierta funcionalidad mucho más simple. Por ejemplo, en el uso del PostScript, el operador `showpage` puede ser sobreescrito con uno personalizado que se aplica a cierto estilo de página, en vez de tener que definir a un operador personalizado o al código de repetición para generar el estilo.