Carne: 2025072099

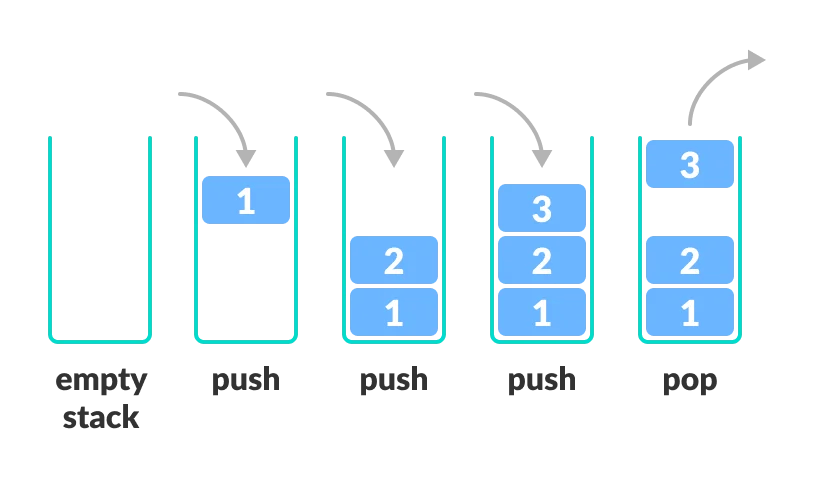
Grupo: 02

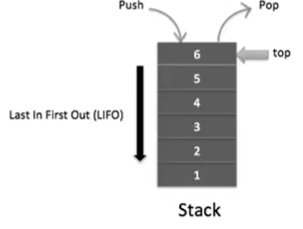
Estudiante: Ronny Espinoza Cordero

Aula invertida #5 Pilas && Colas

**Pilas**

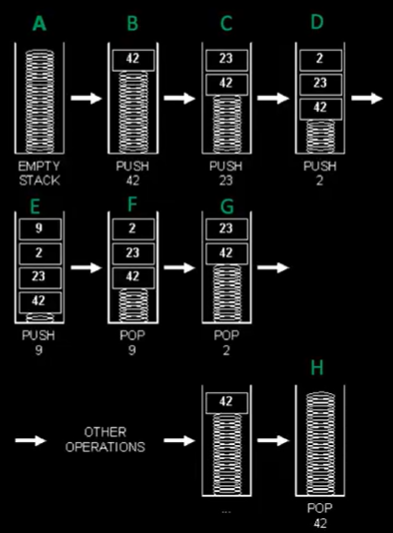
También llamadas stacks. Son un caso especial de lista enlazada donde todos los elementos entran y salen por el mismo lado, de esta forma el primer elemento está en el fondo, el segundo esta arriba del primero y así con todos los elementos. Se puede entender con la analogía de una pila de cajas, en estas la persona no agarra la caja que esta más abajo, agarra la que esta más arriba y cuando se va a agregar una caja esta también se coloca arriba, no abajo

En las pilas suceden operaciones push, top pop. El push es para agregar elementos, aquí se empujan hacia abajo los elementos que ya estaban en la pila. El top se consulta cual es el elemento que está en el inicio (más arriba) El pop es para sacar elementos de la pila, en esta operación ocurre un desenlace entre el elemento que este más arriba y la lista, el elemento no es destruido simplemente sale de la pila.



Un ejemplo de operación push y pop respectivamente fueron los ejemplos de AgregarInicio(Elem) y los recorridos que se hacían con Aux para las diferentes funciones. Estos ejemplos fueron vistos en el resumen de la clase anterior a esta. Para el caso del top en la lección pasada este devolvería la dirección del puntero lista, o el mismo puntero.

**Pilas en un algoritmo funcional**

Para analizar las pilas de mejor forma se verá su aplicación en un código. Primeramente se puede ver como se llama cada función y su representación gráfica de cómo trabajan estos algoritmos.

A. CrearPila(PilaEnteros)

B.PushPila(PilaEnteros,42)

C.PushPila(PilaEnteros,23)

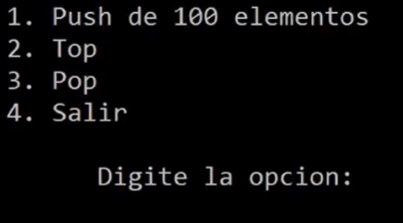
D. PushPila(PilaEnteros,2)

E. PushPila(PilaEnteros,9)

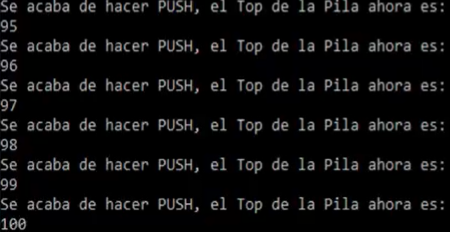
F. X= PopPila(PilaEnteros,2)

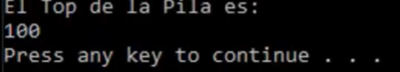
G. Y= PopPila(PilaEnteros,9)

H. Z=PopPila(PilaEnteros)

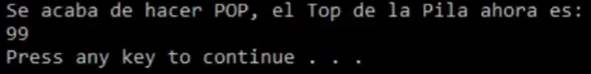
Luego de iniciar el programa y seleccionar la opción 1. Usar Pila, se muestra lo siguiente:

Al seleccionar la opción 1, se hará un push de 100 elementos, uno encima del otro a medida que aumentan, aquí el top va cambiando a medida que se agregan elementos nuevos a la pila, ya que el top siempre es el elemento más nuevo ósea el que está más arriba, esto se puede ver así en el menú:

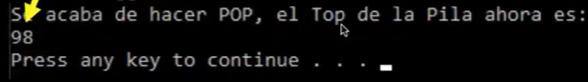


Al seleccionar la opción 2, se mostrará el top de la pila, recordemos que en este momento es 100 ya que la función se ejecuta después de haber usado la pila. El resultado de esta operación se muestra así:

Al seleccionar la opción 3, ocurre un Pop en el que el ultimo elemento de la pila (100) es desenlazado de la lista. Al ejecutar esta operación se muestra lo siguiente:



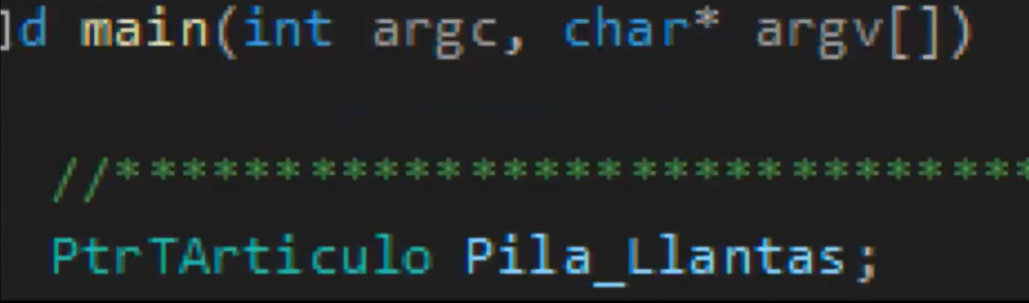
Si se vuelve a ejecutar este proceso el resultado ahora se mostrara así:



Por último al seleccionar la opción 4, se borra la memoria utilizada en el programa, ósea la memoria ocupada por la lista y se cierra el programa.

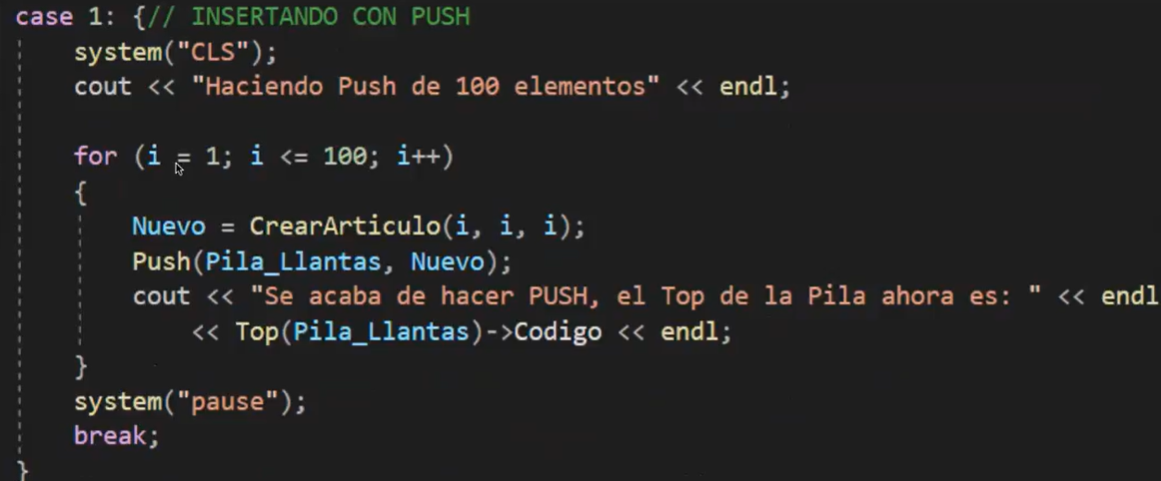
De momento solo se ha analizado la representación grafica de estos procesos y como se muestran en pantalla, ahora vamos a analizar el código que hay detrás de cada proceso.

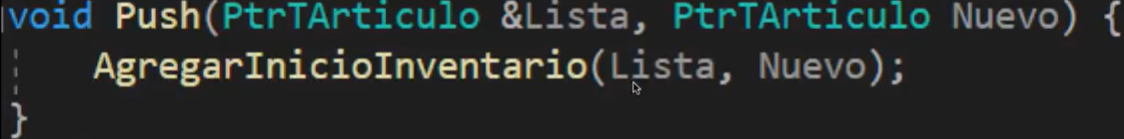
Primeramente en el main se genera la variable Pila\_Llantas de tipo PtrTArticulo(conocido ya en el resumen anterior)



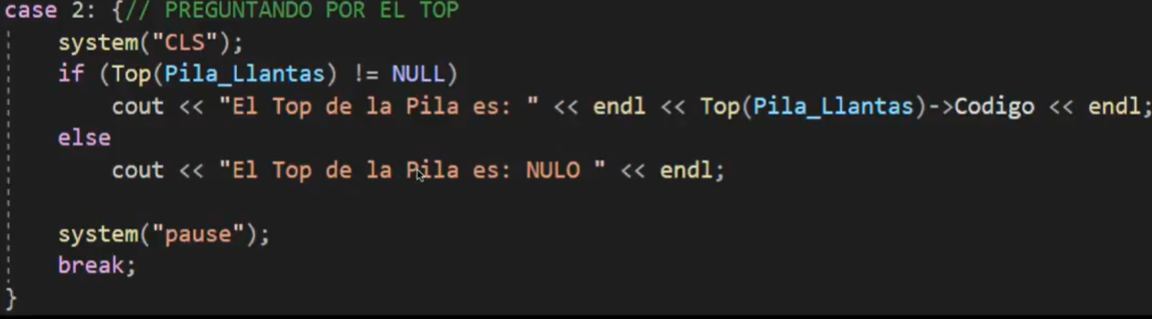
Luego de esto también se utiliza InicializarInventario(Pila\_Llantas), esta función funciona igual que en el resumen anterior. Las funciones nuevas son las que van a requerir más explicación.

Luego de la función del menú se pasa a la funcionalidad de la primera opción, esta es la siguiente:

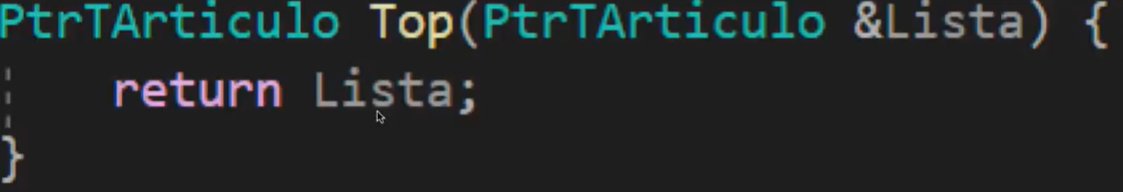


Aquí es importante recordar que la función de CrearArticulo devuelve la dirección del nuevo elemento creado. Además siempre que se pasa por un elemento este se imprime usando la función Top que se analizara a continuación. Un elemento nuevo es la función Push sin embargo esto en realidad es algo engañoso ya que como se puede ver a continuación, esta función no se hace nada, sino que llama a una función ya conocida.

Ahora pasamos al caso 2, para obtener el top de la lista.

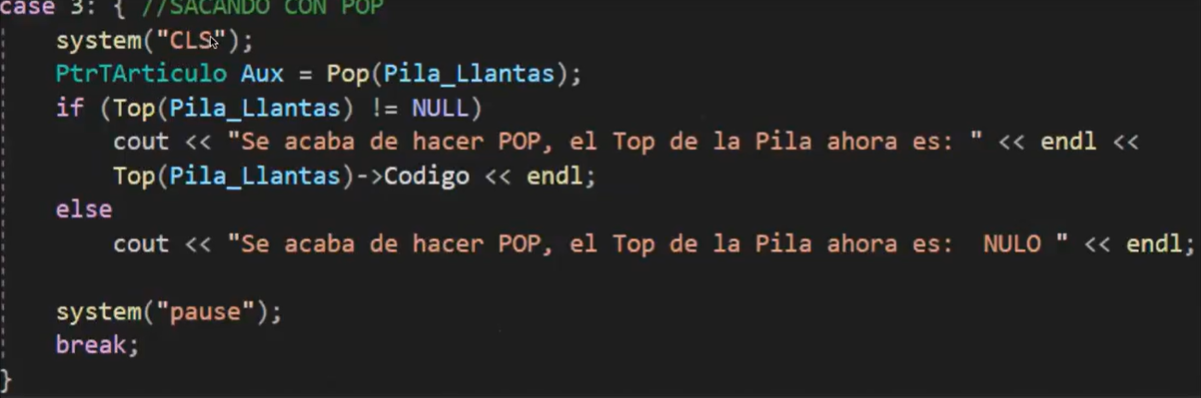


La función analiza que la Pila tenga elementos ósea que no sea NULL. El elemento más particular aquí es la misma función Top.

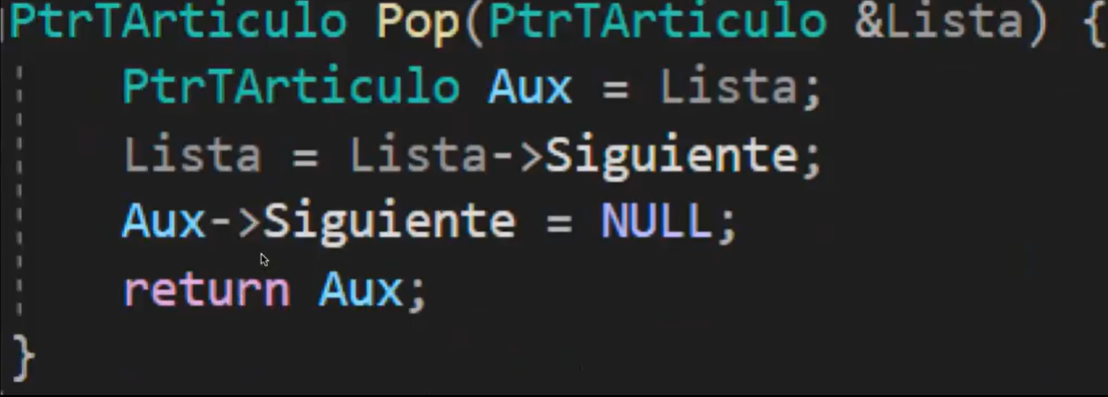


Esta funcion recibe la lista enlazada y retorna el puntero o la coordenada que este guardado en el parámetro lista.

Por ultimo se llega al caso 3, la operación pop.



Aquí primero se le asigna a Aux el Pop de la pila de llantas. Luego se muestra el nuevo valor que tiene el Top, esto se hace verificando que la pila no este vacía y luego mostrando el Top con la función ya vista. Ahora analicemos la operación Pop:



En esta función Aux obtiene el valor de Lista, luego Lista pasa a apuntar al elemento siguiente, el segundo de la antigua lista y el nuevo primer elemento, seguido de esto Aux apunta a NULL produciendo el desenlace y finalmente se devuelve el valor de Aux, es decir, el valor que tenia la lista en un principio.

Cada línea de este código se puede ver como un paso individual. Algo así:

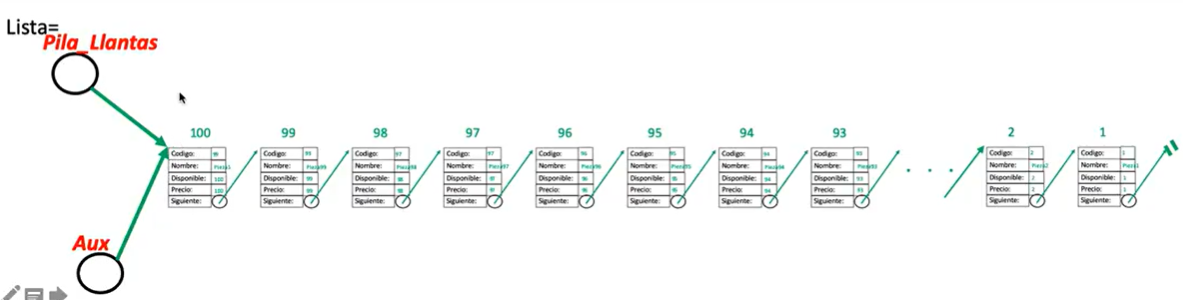
A. PtrTArticulo Aux = Lista;

B. Lista = Lista->Siguiente;

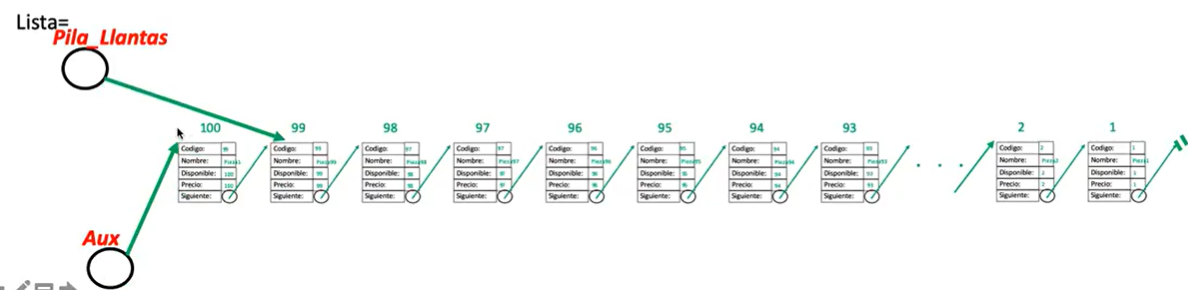
C. Aux->Siguiente = NULL;

D. return Aux;

Vamos a usar esto para describir la función paso por paso.



En el paso A, Aux pasa a apuntar a Lista, Lista en ese momento corresponde al elemento TOP que es 100.



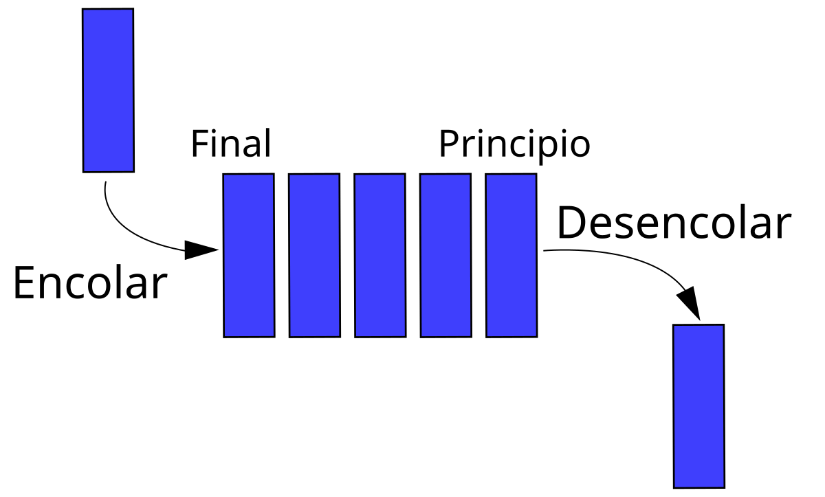
Ahora que se tiene almacenado el valor inicial de lista en Aux se puede proceder al paso B donde Lista ahora apunta al que en un principio era el segundo elemento de la lista.



Por último en los pasos C y D, el elemento Aux se separa de la lista y es retornado.

**Colas**

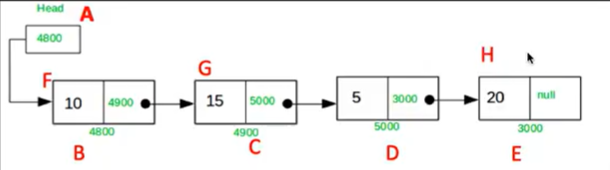
También conocido como un QUEUE. Es un caso especial de lista enlazada, pero a diferencia de las pilas aquí los elementos entran por un lado (el final) y salen por el otro (el inicio), además de esto se tiene que el elemento recién agregado va a estar al final de la estructura. Se puede entender mejor con una analogía en la que ciertas personas están haciendo fila en el banco, la persona que llego de primero será atendida primero, la segunda persona en la fila pasaría a estar al principio cuando el primero sea atendido y así con todas las personas en la fila, si llega alguien este tendrá que esperar a que pasen todos los que están al frente.

En el caso de las colas suceden operaciones encolar, primero y descolar. El encolar es para agregar un elemento al final de una estructura, el primero consulta el elemento que se encuentra al principio en la fila y descolar toma el elemento, lo saca o lo separa de la lista y posteriormente pone al elemento que estaba como segundo en la posición del primero.

Un ejemplo de operación encolar y desencolar respectivamente fueron los ejemplos de AgregarFinal(Elem) y los recorridos que se hacían con Aux para las diferentes funciones. Para el caso del primero en la lección pasada este devolvería la dirección del puntero lista, o el mismo puntero, este corresponde al primer elemento. En general las operaciones son muy parecidas a las de las pilas solo cambiando las posiciones en donde se ejecutan.

**Colas en un algoritmo funcional**

Para analizar las colas de mejor forma se verá su aplicación en un código. Primeramente se puede ver como se llama cada función y su representación gráfica de cómo trabajan estos algoritmos, la representación es una sola imagen en la que cada letra esta en el lugar donde se ejecuta, de esta manera se puede entender de una forma más simple.

A. CrearCola(ColaEnteros);

B. Encolar(ColaEnteros,10);

C. Encolar(ColaEnteros,15);

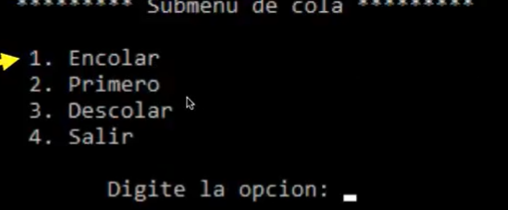
D. Encolar(ColaEnteros,5);

E. Encolar(ColaEnteros,20);

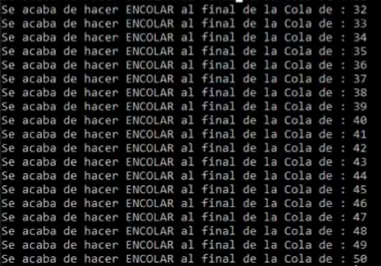
F. X= Descolar(ColaEnteros);//10

G. Y= Descolar(ColaEnteros);//15

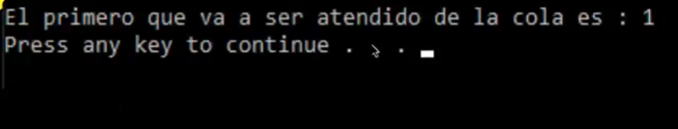
H. Z= Descolar(ColaEnteros);//20

Luego de iniciar el programa y seleccionar la opción 2. Usar Cola, se muestra lo siguiente:

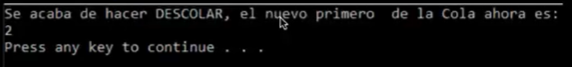
Al seleccionar la opción 1 se producirá algo parecido al caso de las pilas pero aplicado a la operación encolar y con 50 elementos:



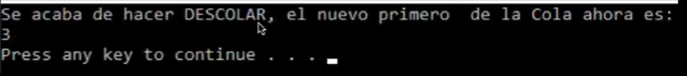
Al seleccionar la opción 2, se mostrar el Primero, que en este caso es el elemento que este más adelante en la cola, recordando que ya se ejecutó la operación 1 anteriormente.



Al seleccionar la opción 3, ocurre un Descolar en el que el primer elemento de la pila (1) es desenlazado de la lista. Al ejecutar esta operación se muestra lo siguiente:



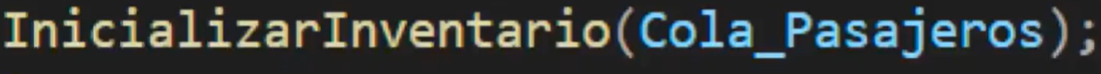
Si se vuelve a ejecutar este proceso el resultado ahora se mostrará así:



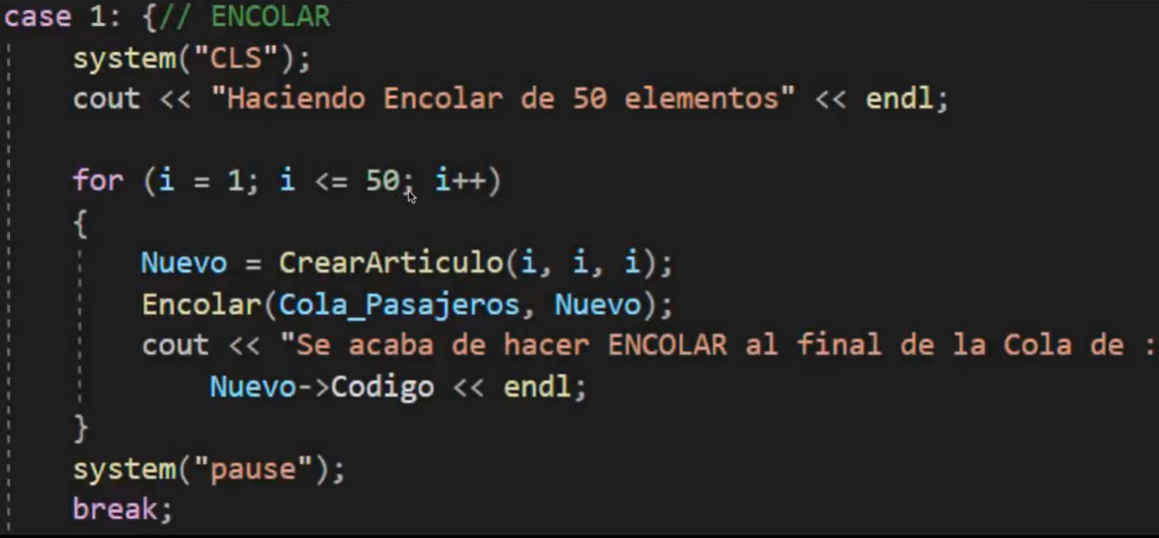
Por último al seleccionar la opción 4, se borra la memoria utilizada en el programa, ósea la memoria ocupada por la lista y se cierra el programa.

Ahora analizando el código que está detrás de todos estos procesos se puede encontrar lo siguiente:

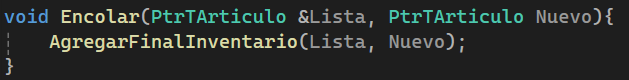
En el main se inicializa el inventario con la función ya conocida:



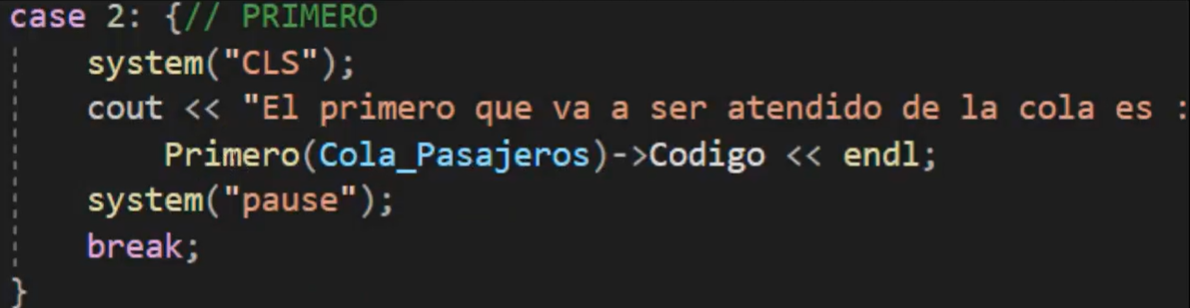
Luego de la función del menú se pasa a la funcionalidad de la primera opción, esta es la siguiente:



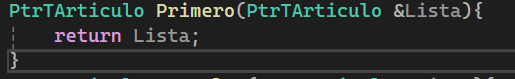
Aquí para encolar un elemento se llama la función Encolar, esto se analizara a continuación pero de igual forma no tiene mucha explicación, solo hace un llamado a la función AgregarFinalInventario, además siempre se imprime el nombre del nuevo elemento, sin necesidad de usar la operación Primero.



Ahora pasamos al caso 2, para obtener el primero de la lista.



Esto funciona con él la función Primero que se entiende así:

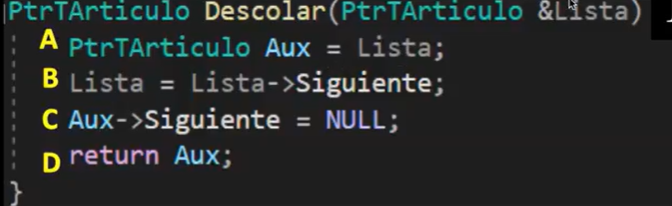


Esta función recibe la lista enlazada y retorna el puntero o la coordenada que este guardado en el parámetro lista.

Por último se llega al caso 3, la operación descolar.



Aquí primero se le asigna a Aux el Descolar de la Cola\_Pasajeros. Luego se verifica que haya elementos en la cola, si todavía los hay se imprime el valor de ese elemento como nuevo primer elemento.



En esta función Aux obtiene el valor de Lista, luego Lista pasa a apuntar al elemento siguiente, el segundo de la antigua lista y el nuevo primer elemento, seguido de esto Aux apunta a NULL produciendo el desenlace y finalmente se devuelve el valor de Aux, es decir, el valor que tenía la lista en un principio.

Cada línea de este código se puede ver como un paso individual. Algo así:

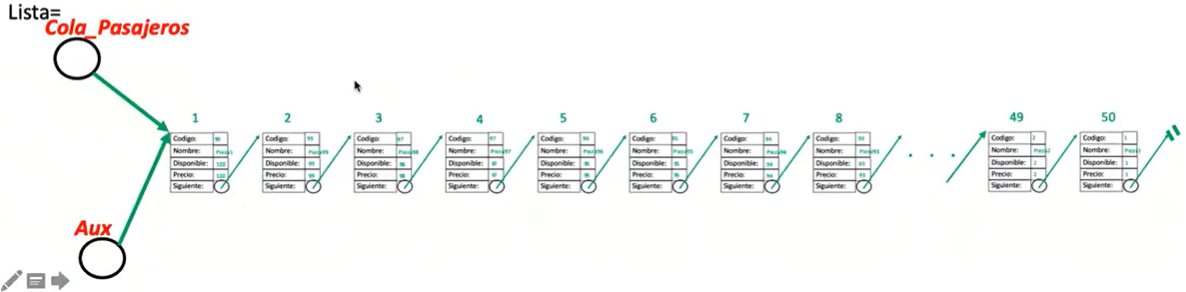
A. PtrTArticulo Aux = Lista;

B. Lista = Lista->Siguiente;

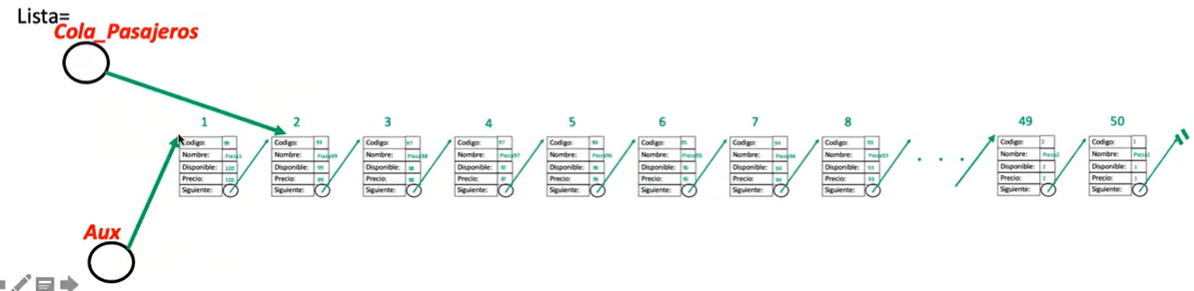
C. Aux->Siguiente = NULL;

D. return Aux;

Vamos a usar esto para describir la función paso por paso.



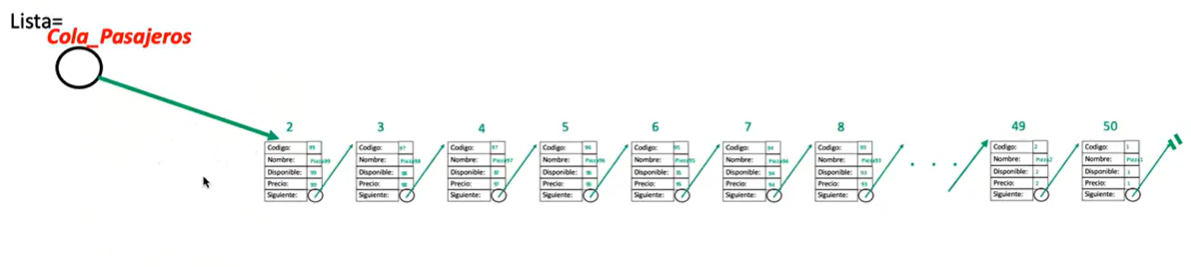
En el paso A, Aux pasa a apuntar a Lista, Lista en ese momento corresponde al elemento Primero que es 1.



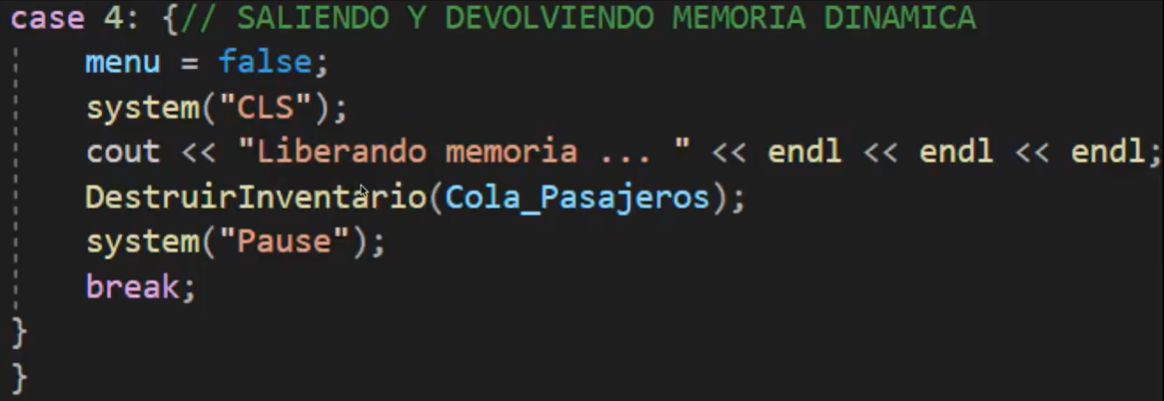
Ahora que se tiene almacenado el valor inicial de lista en Aux se puede proceder al paso B donde Lista ahora apunta al que en un principio era el segundo elemento de la lista.



Por último en los pasos C y D, el elemento Aux se separa de la lista y es retornado.



Finalmente y al igual que en con las pilas, con la opción 4 se libera la memoria destruyendo el inventario.



**Uso de operaciones de pilas**

**Las operaciones push,pop y top aplicado a expresiones algebraicas.**

Normalmente se conocen las expresiones de tipo infijo en la que se presentan operaciones como A+B, aquí el operador (+) esta en medio de dos operandos (A B).

Por otro lado existen las expresiones posfijas donde hay operaciones como A B +, aquí se le da prioridad a almacenar los operandos y luego al operador, de esta se memorizan los dos operandos en una pila y luego cuando aparece el operador salen los operadores.

Un punto importante aquí es la tabla de prioridades que se usa siempre en la matemática. Esta tabla define que operación se debe resolver primero.

|  |  |
| --- | --- |
| Prioridad | Operador |
| 1 | “(” o “)“ |
| 2 | “\*” o “/“ |
| 3 | “+” o “-“ |

|  |  |
| --- | --- |
| Infija | Posfija |
| A+B | A B+ |
| A\*B/C | A B\*C/ |
| A-B/C | A B C/- |
| A+B \*C-D/E | A B C\*+D E/- |

Basado en esta tabla de producen los dos tipos de operaciones. A continuación unos ejemplos:

Esto corresponde al concepto principal del tema.

**Convertir cadena de caracteres en infijo a una cadena de caracteres posfija.**

Este proceso consiste en una descomposición de la expresión, separándola en elementos que luego forma una cadena posfija. Esto se puede analizar con la siguiente tabla en la que se analizara la expresión (A/ (B-C)) \* (D + E)

|  |  |  |
| --- | --- | --- |
| Símbolo | Pila de operadores | Cadena de caracteres |
| ( | ( |  |
| A | ( | A |
| / | (/ | A |
| ( | (/( | A |
| B | (/( | A B |
| - | (/(- | A B |
| C | (/(- | A B C |
| ) | (/ | A B C- |
| ) |  | A B C -/ |
| \* | \* | A B C -/ |
| ( | \*( | A B C -/ |
| D | \*(+ | A B C -/D |
| + | \*(+ | A B C -/D |
| E | \*(+ | A B C -/D E |
| ) | \* | A B C -/D E + |
|  |  | A B C -/D E + \* |

En la columna Símbolo se ponen los caracteres de la expresión en posfijo, en la Pila de operadores se colocan los operadores y en la Cadena de caracteres se escribe la conversión. Un punto importante es que los operandos pasan directamente a la tercera columna mientras que los operadores solo pasan a esta columna cuando se cierra un paréntesis, pasando los elementos que vayan a quedar dentro de ese paréntesis o cuando ya no hay elementos.

Lo que ocurre en la segunda columna se puede ver como una operación push de pila.

**Operaciones con expresiones posfijas**

Finalmente y para explicar cómo función las operaciones en posfijo vamos a analizar ejemplos además de un caso con números reales y aplicándolo a pilas.

Proceso:

ABC-/DE +\*

|  |  |
| --- | --- |
| Operación | Cadena de caracteres |
| R1 = B C - | A R1/DE +\* |
| R2 = A R1 / | R2 DE +\* |
| R3 = D E + | R2 R3\* |
| R4 = R2 R3 \* | R4 |

Esto aplicado a una pila donde A= 8, B= 2, C= 4, D=3, E=9, ocurre así.



Todo esto se hace haciendo un push de los elementos que hay en la expresión y si hay un operando este se ejecuta.