Imagen que contiene Texto

Descripción generada automáticamente

**Instituto Tecnológico de Costa Rica**

**Escuela de Ingeniería en Computación**

**Campus Tecnológico Central de Cartago**

**[IC2001] Estructuras de Datos**

**Profesor: Ing. Víctor Garro Abarca**

**Manual de técnico**

Calculadora

José Daniel Araya Ortega c.2022209303

Jocsan Adriel Pérez Coto c.2022437948

Entrega: 7/11/2022

II Semestre

Contenido

[Estructuras de datos 3](#_Toc118684843)

[Nodos 5](#_Toc118684844)

[Relación entre estructuras 6](#_Toc118684845)

[Algoritmos/Lógica 6](#_Toc118684846)

[Main 14](#_Toc118684847)

# Estructuras de datos

**Lista enlaza simple:** Para contener de manera dinámica el número de posición de una pista del programa, se tiene que aplicar una lista enlazada simple, que nos permite agregar o eliminar dinámicamente los valores necesarios cuando el programa lo requiera, gracias a que trabaja con punteros y variables anónimas. Además de que esta se genera a base structs, lo que nos permite generar nuestros elementos con los parámetros que necesitemos, generando los nodos de la lista.

Un ejemplo de cómo se ve una lista enlazada es el siguiente:

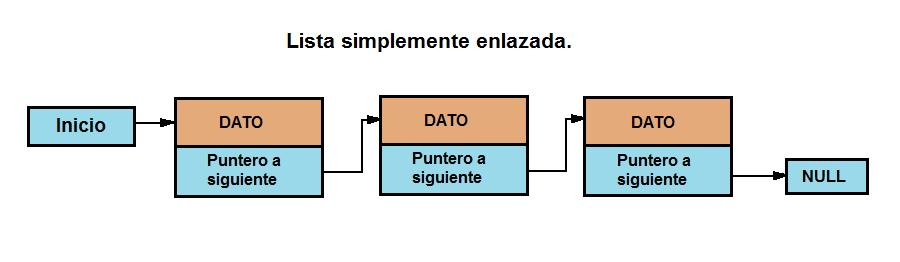


Fig 1. Lista enlazada simple. Tomado de: https://analisisyprogramacionoop.blogspot.com/2017/07/lista-simplemente-enlazada-C-sharp.html

**Cola (FIFO):** En este proyecto se implementará tanto una cola de elementos, donde se debe primero encolar, y luego desencolar de manera dinámica los elementos, como estructura que asemeja este comportamiento, pero no utilizan funciones de encolamiento y desenconamiento.

Para el caso de la cola completa esta estará vacía y con forme se le encolan elementos, se le irán también desencolando otros. Para comprender este concepto en la real tenemos:



Fig 2. Ejemplo real de FIFO. Tomado de: https://www.freepik.es/fotos-vectores-gratis/fila-de-personas

También lo podemos ver como una lista enlazada:

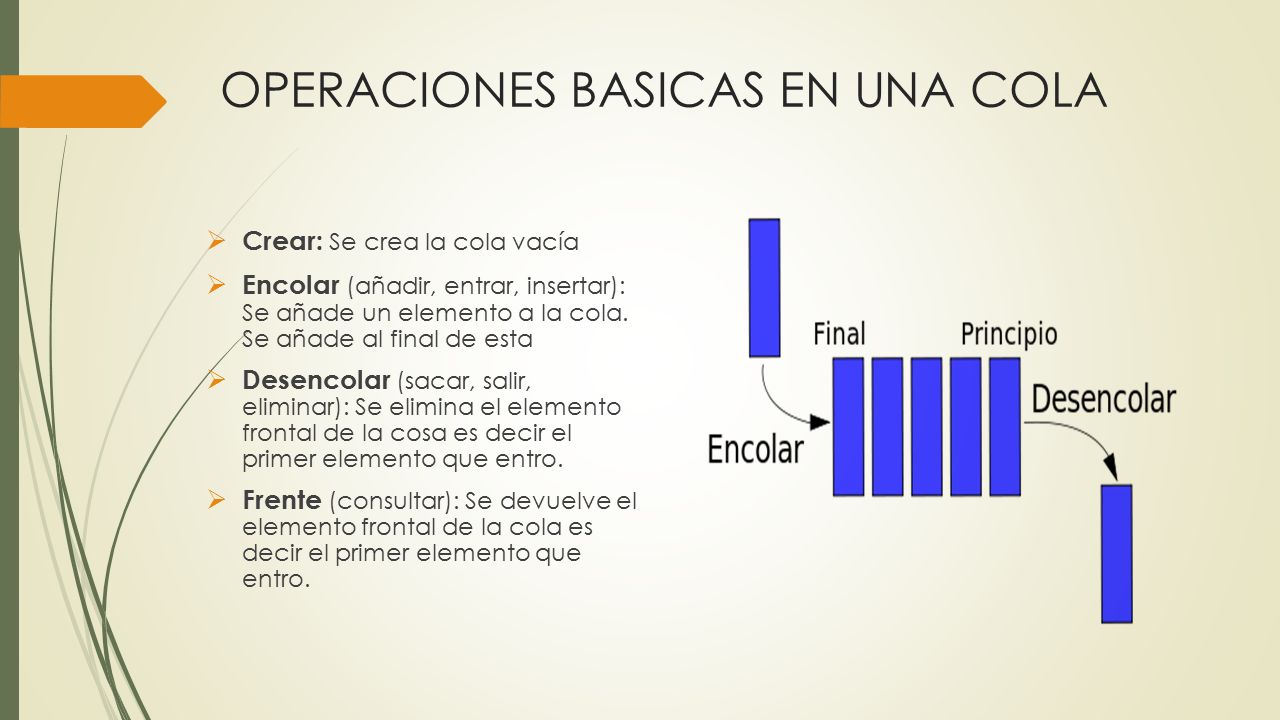


Fig 3. Ejemplo en LE de FIFO. Tomado de: https://slideplayer.es/slide/16359303/

Por otro lado, este comportamiento se puede generar en listas estáticas donde se recorre la lista y en cada elemento se realiza alguna acción, donde al terminar de recorrer y volver a recorrer el primer en la lista será el primero en ser leído y el ultimo al final. Para utilizarlo y comprenderlo mejor hay que pensar en una fila circular, donde nadie sale y se recorre continuamente, como una lista enlazada circular. Recordando siempre que no es una lista enlazada de este tipo pero que se asemeja en comportamiento.

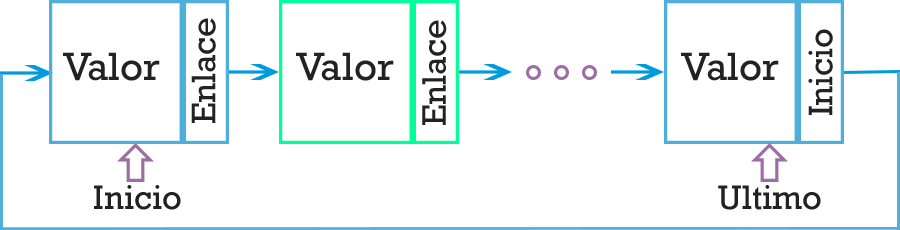


Fig 4. Ejemplo de estructura de LE circular. Tomado de: http://codigolibre.weebly.com/blog/listas-circulares-simples-en-java

# Nodos

Para el proyecto se utilizarán dos tipos nuevos de datos:

**grupo:** Nodo principal que se utilizara para todas las operaciones.

Texto

Descripción generada automáticamente

Los parámetros son:

* **char simbol:** Este es el char que en caso de ser un nodo operador se tomara en cuenta para el análisis.
* **int valor:** Este es el valor que se utiliza para cuando se trabaje con números, o se tenga que evaluar la prioridad en caso de ser un nodo operador.
* **int esOperados:** Este booleano lo utilizamos para saber si el nodo actúa como un operador o como un valor o símbolo deferente de estos.
* **bool operacion:** Debido a la presencia de divisiones se utilizará este parámetro en la evaluación de resultados.
* **Nodo\* Siguiente:** Puntero del mismo tipo para la lista enlazada.

# Relación entre estructuras

Nuestra lista enlazada simple se relacionará con los elementos de tipo PtrNodo, donde estos serán los nodos de esta. Esta lista en relación con el programa se utilizará como un registro dinámico de los valores ocupados en el programa.

# Algoritmos/Lógica

Pilas que se utilizaran en los diversos algoritmos, siendo la de operadores, la de salida de expresiones y la que se utilizara para evaluar las instrucciones.

**Texto

Descripción generada automáticamente**

**Manejo de listas enlazadas:**

* Inicializa una lista enlazada, tomada de ejemplos antiguos.

**Texto

Descripción generada automáticamente**

* Destruye una lista enlazada, tomada de ejemplos antiguos.

**Texto

Descripción generada automáticamente**

**esNum:** Esta devuelve true si el char pasado corresponde a un número, según la tabla ASCII, false de lo contrario.

**Texto

Descripción generada automáticamente**

**esOpe:** Esta devuelve true si el char pasado corresponde a un operador de la calculadora, según la tabla ASCII, false de lo contrario.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**ListaEnlazadaToString**: Toma una pila(lista enlazada), y devuelve como un string con espacios entre sus símbolos, esto se utilizará a la hora de convertir una

expresión infija a una posfija. Esta se utiliza cuando la pila no tiene que tomar en cuenta valores numéricos.

Texto

Descripción generada automáticamente

**ListaEnlazadaToString2:** Toma una pila(lista enlazada), y devuelve como un string con espacios entre sus componentes, esto se utilizará a la hora de convertir una

expresión infija a una posfija y evaluar el valor de sus variables. Esta se utiliza tomando en cuenta que los nodos que no sean operadores se agregara su valor.

Texto

Descripción generada automáticamente

**CrearNodo**: Crea y retorna un nodo, donde si es operador o no se decidirá en base a char pasado. Si no es un operador se igualan los valores pasados y se declara esOperador como false.

Texto

Descripción generada automáticamente

**AgregarInicio:** función que recibiendo dos referencias una de una lista y una de un nuevo objeto, agrega al inicio de la lista el objeto nuevo.

Texto

Descripción generada automáticamente

**Push:** Dada una pila y un objeto, hace un agregar al incio.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Top:** Dada una pila, retorna, sin desligar, el primer nodo de la pila, el tope de esta.

Texto

Descripción generada automáticamente

**Pop:** Dada una pila, retorna, desligando, el primer nodo de la pila.

Texto

Descripción generada automáticamente

**covertirInPos**: Esta función toma un string asumido en infijo y retorna un string que represente la misma fórmula, pero en formato posfijo. Se recorre el string paso por paso.

Texto

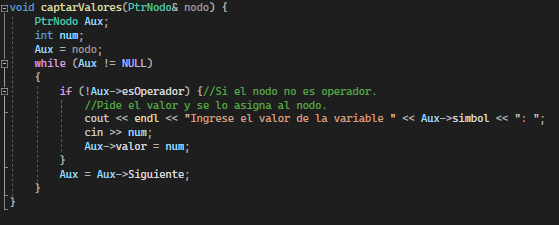
Descripción generada automáticamente

**EvaluacionPosFija**: Esta función toma un string asumido en posfijo y evalúa sus valores, para devolver un valor de resultado

Texto

Descripción generada automáticamente

**captarValores**: Esta función recibe una pila, y donde los nodos que no sean operadores pide por consola un valor para asignarles



# Main

Para esta parte solo se enfocará en las partes más importantes del main, como lo son el llamado a las funciones necesarias para la evaluación de la cifra indicada.

Texto

Descripción generada automáticamente

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Se crea un ciclo de un menú con diferentes opciones**

**Texto

Descripción generada automáticamente**

**opc:** va a cambiar según lo que se indique

**Caso 1:** luego de indicar el numero 1, se llama a la función de convertirInPos

Texto

Descripción generada automáticamente

Al final, se va a entrar un resultado según lo que indique el usuario.

**Caso 2:** luego de indicar el número 2, se llama a la función de EvaluacionPosFija

**Texto

Descripción generada automáticamente**

Al final, se va a entrar un resultado según lo que indique el usuario.

**Caso 3:** luego de indicar el número 3, se llama a la función de convertirInPos

**Texto

Descripción generada automáticamente**

En el caso 3, luego se van a pedir los valores necesarios según lo que indique el usuario.

Al final de cada caso siempre se va a destruir la pila, ya que esta no se va a necesitar mas y se necesita que este limpia para otro caso

**Caso 4**: Funciona para salirse del programa.