Documentación Práctica 2
Estructura de Datos Lineales

Introducción.

Una pila es un tipo de dato abstracto con dos primitivas llamadas Push, la cual inserta elementos, y Pop que extrae elementos. Se le llama pila ya que al hacer uso de Push este guarda los elementos uno sobre otro, apilando uno tras otro. El orden en el que los elementos son extraídos de la pila es que siempre extrae el último elemento que se insertó, es decir el elemento que está en el tope, también es llamado LIFO (Last Input First Output).

Descripción del Problema.

De un archivo de texto y con ayuda de métodos integrados en C++ se extraerá una operación aritmética, esta operación se guardará en un arreglo del tamaño de la operación. Posteriormente se guardará cada elemento de ese arreglo en una lista ligada.

Con las funciones de extraer e insertar de la lista se manipulará la lista que contiene la operación y con ayuda de una pila se realizará un método que pase la operación aritmética de su forma infija a la forma postfija. La operación en su forma infija se ve de la siguiente manera: 23*53+21-43, en su forma infija se vería de la siguiente manera: 23 53 * 21+ 43 -. También se realizará un método que resuelva la operación en su forma postfija.

La función que convierte la operación aritmética de su notación infija a postfija deberá considerar si la operación posee paréntesis y resolver con orden de precedencia, es decir si la operación es 2+3*5 el método deberá realizar la operación de manera que primero se resuelva el 2*5 y posteriormente se sume el 2. Si los métodos son correctos el programa deberá mostrar el resultado correcto.

Desarrollo.

A continuación, se mostrarán los pseudocódigos de los métodos que se implementarán para resolver el problema.

Pseudocódigo estructura nodo:

- NODO:
  carácter chDato
  entero iDato
  Nodo pnSig.

Pseudocódigo Lista:

- Insertar nodo al final:
  Entrada: I, E
  Salida: I
  Si (I = 0) entonces
    I = E
  Otro
    E.sig = I
    I = E
  Fin Si
- Crear Nodo Carácter:
  Nodo Aux
  Aux <- nuevo (nodo)
  Aux.chDato <- chDato
  Aux.iDato <- NULO
  Aux.sig <- 0
- Crear Nodo Entero:
  Nodo Aux
  Aux <- nuevo (nodo)
  Aux.iDato <- iDato
  Aux.chDato <- NULO

Aux.sig <- 0

- Extraer Nodo:
  Entrada: I, E
  Salida: E
  Si (I = 0) entonces
    Regresa (0)
  Aux <- I
  I <- Aux.Sig
  Aux.Sig <- 0

- Inicio Lista
  Nodo Inicio
    Regresa I

Pseudocódigo PILA:

- Estructura Pila:
    Nodo contador <- 0
    Nodo tope <- 0
- Método PUSH:
  Entrada: Tope, E
  Salida: Tope
  Si (Tope = 0) entonces
    Tope <- E
  Otro
    E.Sig <- Tope
    Tope <- E
  Fin Si
- Método POP:
  Si (Tope = 0) entonces
    Regresa(0)
  Otro
    Aux <- Tope
    Tope <- Aux.Sig
    Aux.Sig <- 0
    Regresa (0)
  Fin Si

Pseudocódigo Precedencia:

- Entrada: carácter operador

```
Salida: entero
Según Sea (operador) Entonces
Caso '(':
  Regresa (-2)
Caso ')':
   Regresa (-1)
Caso '+':
Caso '-':
  Regresa (1)
Caso '*':
Caso '/':
   Regresa (2)
Caso '^':
  Regresa (3)
Default: Regresa (0)
Fin Según
```

Pseudocódigo Infijo a Postfijo:

-
```
Entrada: listaInfija
Salida: listaPosfija.Inicio
Si (listaInfija.Inicio = NULO)
  Regresa (0)
Otro
  Mientras(listaInfija <> 0) Hacer
    Aux <- listaInfija.ExtraerNodo
    precedenciaAux <- precedencia(Aux.chDato)
    Si (Aux.iDato) entonces
      listaPosfija.InsertarFinal(Aux)
    Sino Si (precedenciaAux = -2)
      Pila.Push(Aux)
    Sino Si (precedenciaAux = -1)
      i <- precedencia (Pila.TopePila.chDato)
      Mientras (Pila.TopePila <> NULO y i <> -2) Hacer
        AuxPila <- Pila.TopePila
        Pila.POP
        Si (precedencia(AuxPila.chDato) <> -2) entonces
          listaPosfija.InsertarFinal(AuxPila)
        Fin Si
      Fin Mientras
```

```
        Si ( i = 2) entonces
          AuxPila <- Pila.TopePila
          Pila.POP
        Fin Si
      Otro
        Si (Pila.TopePila <> NULO)
          preAuxTope <- precedencia(Pila.TopePila.chDato)
        Fin Si
        Mientras(Pila.Tope Pila<>NULO y precedenciaAux<= preAuxTope) Hacer
          AuxPila = Pila.TopePila
          Pila.POP
          listaPosfija.InsertarFinal(AuxPila)
        Fin Mientras
        Pila.PUSH(Aux)
      Fin Si
    Mientras (Pila.TopePila <> NULO)
      AuxPila = Pila.TopePila
      Pila.POP
      listaPosfija.InsertarFinal(AuxPila)
    Fin Mientras
  Fin Si
  Aux = listaPostfija.Inicio
  Mientras(Aux) Hacer
    Si (Aux.iDato) entonces
      Imprime Aux.iDato
    Otro
      Imprime Aux.chDato
    Fin Si
    Aux = Aux.Sig
  Fin Mientras
  Regresa (listaPosfija.Inicio)
```

Pseudocódigo Evalúa Notación Posfija:

- Entrada: nodo Lista
  Salida: TopePila

```
  Si (Lista) Entonces
    Mientras (Lista <> 0) Hacer
```

```
Lista <- Lista.Sig
Aux = listaPosfija.ExtraerNodo
Si (Aux->iDato) entonces
  PilaOperacion.PUSH(Aux)
Otro
  pilaAux<- PilaOperacion.POP
  pilaAux2<-PilaOperacion.POP
  Según Sea (Aux->chData) entonces
    Caso '+':
       pilaAux.iDato += pilaAux2.iDato
       PilaOperacion.PUSH(pilaAux2)
     Caso '-':
       pilaAux2.iDato -= pilaAux.iDato
       PilaOperacion.PUSH(pilaAux2)
     Caso '*':
       pilaAux.iDato *= pilaAux2.iDato
       PilaOperacion.PUSH(pilaAux)
     Caso '/':
       pilaAux2.iDato /= pilaAux.iDato
       PilaOperacion.PUSH(pilaAux2)
     Caso '^':
       iNum = pilaAux.iDato
       Mientras (pilaAux.iDato <> 1) Hacer
         pilaAux2.iDato <- pilaAux2.iDato *iNum
         pilaAux.iDato <- pilaAux.iDato - 1
       Fin Mientras
         PilaOperacion.PUSH(pilaAux2)
      Fin Segun
     Fin Si
```

Pseudocódigo Leer Archivo:

- Entrada: Archivo
  Salida: Cadena operacionLinea
  Archivo.Abrir("Nombre o Dirección de Ubicación de Archivo".txt, Leer)
  Si (Archivo.Abierto = Falso) entonces
    Imprime "No es posible abrir el archivo"
  Otro
    ObtieneLinea(Archivo, operacionLinea)

Imprime “Operacion: ” OperaciónLinea FinalLista
Archivo.Cerrar.
Fin Si
Regresa operacionLinea.

Resultados:
De un archivo de texto se obtiene la operación aritmética en su notación infija, esa operación se obtiene del archivo usando la clase READFILE la cual nos regresa una cadena que contiene la operación. Después hacemos uso de un arreglo de caracteres con la longitud de la cadena regresada por el método anterior.

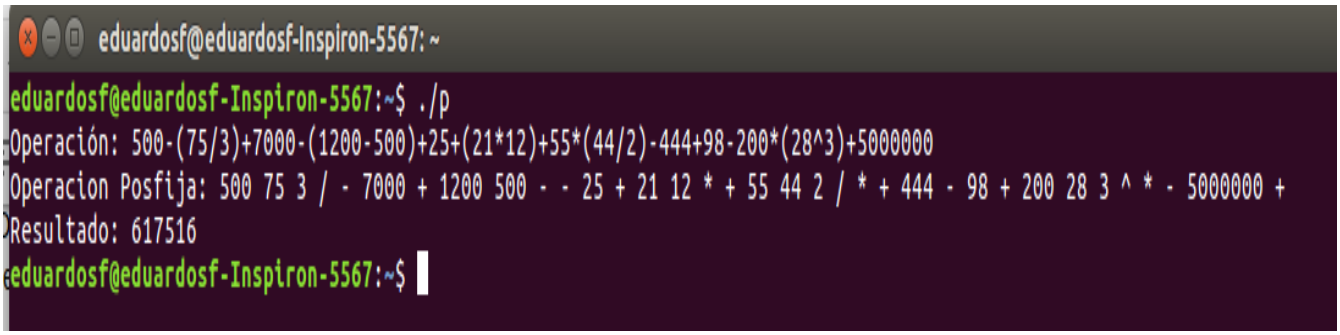
Con el arreglo conteniendo cada elemento de la operación en un índice hacemos uso de un ciclo para, dentro del ciclo para hay una condición “Si” que pregunta si la precedencia del primer elemento del arreglo es cero, esto nos dice si el carácter en el índice es un número y se procede a realizar la conversión de carácter a un entero, después de la condición anterior se realiza otra condición preguntando si el elemento que se encuentra en el índice siguiente es igual a cero o si la precedencia de este es diferente de cero si esto es cierto se crea un nodo de tipo entero que guarda el recién convertido carácter a entero y se inserta al final de la lista. Si el carácter no es un entero se crea un nodo de tipo carácter que guarda el elemento y lo inserta al final de la lista. La lista es una lista ligada simple.

Ya con la lista completa esta se pasa como parámetro a la instancia infix_postfix de la clase INFIJO_POSFIJO para que regrese la operación en su notación infija a notación postfija, la función regresa el nodo inicial de la lista.

Una vez realizada la conversión usamos el nodo regresado y lo usamos en la función evaluaPosfijo de la instancia infix_postfix de la clase INFIJO_POSFIJO para que resuelva la operación en su notación postfija y nos regrese el resultado. Ya evaluada se imprime en la consola el resultado.

A continuación, se muestra una imagen de la consola que muestra:

- La operación aritmética extraída de un archivo en notación infija.
- La operación en notación postfija.
- El resultado de la operación.



Manual de usuario:

- Para compilar el programa es necesario escribir lo siguiente en la terminal de LINUX:
  - **g++ /home/nombreUsuario/ubicaciónArchivo/nombreArchivo.cpp -o p**
- Para ejecutar el program se introduce lo siguiente:
  - **./p**
  - La última letra de la línea de compilación indica la letra que va después del “ **./** ”.
- Para que el programa compile y se ejecute correctamente es necesario que el archivo de texto que contiene la operación se encuentre en la misma carpeta en dónde se encuentra el archivo .cpp del programa.
- También por cuestiones que no pude identificar el método que abre el archivo no puede abrir el archivo solo por el nombre es necesario que se ingrese la ubicación del archivo, por lo cual si se descarga el archivo será necesario que se modifique la línea 257 del código, la cual se encuentra dentro de la clase READFILE y dentro del método readFile, la línea es la siguiente:
  - file.open("/home/eduardosf/EsDatosLineales/Operaciones.txt", ios::in);
  - El texto en color verde es el que se debe modificar con la nueva ubicación del archivo de texto.