****

INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Computo

**Estructura de datos**

Profesor:

Tecla Parra Roberto

Alumna:

Saldaña Aguilar Gabriela

Manual Técnico y de usuario del Proyecto:

“Derivador”

Fecha de entrega: 29/05/14

Índice:

Manual Técnico

Objetivo……………………pg3

Diagrama de Bloques…..pg14

Explicación Bloque 1……pg4

Explicación Bloque 2….…pg4

Explicación Bloque 3……pg7

Manual de Usuario

Modo de uso…………………..pg11

Entrada y Salida de consola...pg13

OBJETIVO:

Se construyó una aplicación la cual nos dará la posibilidad de ayudar en la construcción de una derivada; esto nos simplificará ampliamente la tarea de realizar operaciones y solucionará el problema de que en dado caso y circunstancias que se requiera realizar una operación de manera práctica, podemos recurrir al uso de esta herramienta.

Para generar está aplicación se hizo utilización de las estructuras de datos:

Árbol binario y Pila.

Contenido Bloque1:

En este bloque pedimos al usuario insertar una expresión en posfijo (para mayor información sobre este tema ver Manual de Usuario), la cual empezaremos a leer en el siguiente bloque.

Contenido Bloque 2:

Aquí comienza la lectura de la expresión a derivar la cual se guardará en un árbol de expresión con uso de una pila mientras se está recorriendo la cadena. Se decidió utilizar un ciclo for para leer cada elemento de la cadena, así podemos ir separando de la siguiente manera:

Caso1:

Si se llega a encontrar con un número, mandaremos a llamar una función la cual nos creará un NodoArb que contenga solamente un número en la parte del dato. Este nodo se guarda en la pila.

Caso 2:

Si llega a encontrar uno de los siguientes símbolos: +,-,\*,/,^.

Como para estos símbolos se necesita el uso de dos funciones (mínimo) y una pila se creará un NodoArb al cual se le pasará el símbolo en la parte de la estructura llamada dato y a los extremos se realizará un pop(p) a lo que la pila almacenó previamente, uniendo primero el lado derecho y finalmente el izquierdo.

Caso 3:

Para los siguientes símbolos: s,c,l,e.(Nota: Si quiere saber el significado de estos símbolos ver Manual de Usuario).

Para estos símbolos se requiere el uso de una función por lo que el lado izquierdo del NodoArb va a ir vacio.

De esta manera podemos separar la cadena e ir identificando cada operador para poder crear el árbol de expresión.

Estructuras usadas para el Bloque 2:

|  |  |
| --- | --- |
| struct pila  {  int tope;  void \*\*eltos;  };  typedef struct pila Pila; | struct NodoArb  {  void \*dato;  struct NodoArb \*der;  struct NodoArb \*izq;  };  typedef struct NodoArb NodoArb; |

Funciones utilizadas:

|  |
| --- |
| NodoArb \*creaNodoArb(void \*dato,NodoArb \*der,NodoArb \*izq);  Pila \*creaPila(int n);  void \*push(Pila \*p,void \*dato);  void \*pop(Pila \*p);  char \*creaChar(char c); |

CONTENIDO BLOQUE 3:

Esta es la parte del código que va a evaluar el árbol de expresión para ir derivando ese mismo árbol.

El proceso es el siguiente:

Se pasa a evalDerivada el apuntador del árbol de expresión:

NodoArb\*d=NULL;

d=evalDerivada(pop(p)); //EVALUANDO ÁRBOL DE EXPRESIÓN

Hay que decir que la función de evalDerivada es una función recursiva e indirecta, por tanto, lo que haga en un nodo lo va a hacer en los demás y no para hasta encontrar la condición que detiene la recursividad.

Lo que hace la función es lo siguiente:

\*Primero hay que parar la recursividad para ello revisa que el nodo tratado sea una hoja, si esto se cumple, revisará que tenemos en la parte del dato y regresará:

\*1, si tenemos a la variable x.

\*0, si se trata de una constante.

\*En el caso de que no sea un nodo hoja:

\*Revisa que símbolo se tiene en el dato,

y dependiendo que símbolo sea manda a llamar a su correspondiente derivada.

El caso de los símbolos que tengan que utilizar dos funciones pasamos la parte izquierda (U)

y derecha (V) por separado, las que sólo necesitan una función se les pasa la parte derecha (ver diagramas de árbol) .

Y así recursivamente lo hará ya que en la llamada de las funciones que derivan volvemos a llamar a evalDerivada, por tanto es recursivamente indirecta.

Para ir generando los árboles de las derivadas se crearon una serie de apuntadores a NodoArb para ir uniendo uno con otro y simplificar el código. Al final el árbol resultante es la fórmula necesaria para derivar.

Finalmente se imprime el árbol y dependiendo como lo quieran leer se usará recorrido : Inorden, Posorden y Preorden.

DIAGRAMAS DE ÁRBOL

DE LAS DERIVADAS

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
|  |

**Modo de uso:**

El programa tiene mensajes que lo irán guiando a lo largo de su uso, se pretende que sea de fácil uso y entendimiento, esperando que el usuario se encuentre familiarizado con la notación en postfijo[[1]](#footnote-1) ya que la expresión principal será dada de esa forma, para así poder omitir el uso de paréntesis en la misma.

Antes de comenzar, a utilizar el programa, se presentara la sintaxis utilizada para el manejo de las funciones.

|  |  |
| --- | --- |
| Operador | Símbolo |
| SUMA | + |
| RESTA | - |
| MULTIPLICACION | \* |
| DIVISION | / |
| POTENCIA | ^ |
| SENO | s |
| COSENO | c |
| TANGENTE | t |
| COTANGENTE | T (MAYUSCULA) |
| SECANTE | S (MAYUSCULA) |
| COSECANTE | C (MAYUSCULA) |
| EXPONENCIAL | e |
| LOGARITMO | l |

**Nota: Se toman como operadores a las funciones trigonométricas, logarítmicas y exponenciales, ya que estas operan a una función.**

Algunos ejemplos de cómo poner expresiones en postfijo, al igual que su interpretación en una expresión normal. El primer paso es el poner la expresión escrita en postfijo, por tal motivo es de mucha importancia el tener una idea clara de cómo se utilizan los operadores y la combinación de ellos.

|  |  |
| --- | --- |
| EXPRESIÓN | MODO POSTFIJO |
| x+x | xx+ |
| x-x | xx- |
| 6x | 6x\* |
| 6x/(x+1) | 6x\*x1+/ |
| (6x+2)4 | 6x\*2+4^ |
| Sen(3x2) | 3x2^\*s |
| Cos(3x2) | 3x2^\*c |
| Tan(3x2) | 3x2^\*t |
| Cot(3x2) | 3x2^\*T |
| Sec(3x2) | 3x2^\*S |
| Csc(3x2) | 3x2^\*C |
| e(3x+1) | 3x\*1+e |
| Ln(cos(3x2)) | 3x2^\*cl |

Siguiendo con los ejemplos anteriores también se mostrara el cómo se vería su derivada en salida a pantalla, la cual si no hubo ningún error en la forma de meter los datos, tendría que estar bien.

|  |  |
| --- | --- |
| EXPRESIÓN EN POSTFIJO | DERIVADA PRESENTADA |
| xx+ | 1+1 |
| xx- | 1-1 |
| 6x\* | 0\*x+6\*1 |
| 6x\*x1+/ | 0\*x+6\*1\*x+1-6\*x\*1+0/6\*x^2 |
| 6x\*2+4^ | 0\*x+6\*1+0\*4\*6\*x+2^4-1 |
| 3x2^\*s | cos3\*x^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x2^\*c | -sen3\*x^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x2^\*t | sec3\*x^2^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x2^\*T | -csc3\*x^2^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x2^\*S | sec3\*x^2\*tan3\*x^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x2^\*C | - csc3\*x^2\*cot3\*x^2\*0\*x^2+3\*1\*2\*x^2-1 |
| 3x\*1+e | exp3\*x+1\*0\*x+3\*1+0 |
| 3x2^\*cl | 1/cx^2\*-sx^2\*1\*2\*x^2-1 |

**Nota: El programa no simplifica valores, ni tampoco está implementado el uso de paréntesis, por eso las soluciones se presentan de esta manera.**

Se recomienda que para poder estar seguro de que el programa esté funcionando de manera correcta, usted realice los ejemplos aquí mencionados para que este seguro de que todo está en orden.

Ya que se mostraron algunos ejemplos de cómo realizar la notación de algunas expresiones, ahora usted puede hacer uso de la aplicación para realizar algunos cálculos más complejos o para comprobar cálculos que usted esté realizando.

DIAGRAMA DE BLOQUES:

BLOQUE 1

BLOQUE 3

BLOQUE 2

LECTOR DE CADENA

SALIDA

ÁRBOL DE DERIVADA

ÁRBOL DE EXPRESIÓN

POSFIJO

ENTRADA

1. También conocida como “Notación Polaca Inversa”, es un método algebraico alternativo de introducción de datos. [↑](#footnote-ref-1)