



Instituto Politecnico Nacional



ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”

TEORÍA COMPUTACIONAL

PRÁCTICA 5: GRAMÁTICA LIBRE DE CONTEXTO (GLC)

PROFA: Luz María Sánchez García

ALUMMNO: Rojas Alvarado Luis Enrique

GRUPO: 2CM11

INTRODUCCION

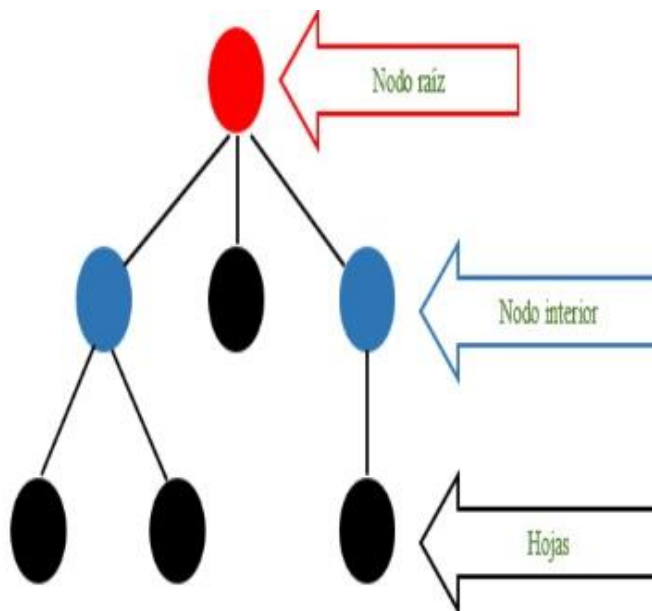
El propósito de ésta práctica consistió en que a partir de ciertas condiciones seleccionadas por el alumno, ya sea balanceo de parentesis, palindromos o algun ejercicio hecho en clase, se requerirá realizar el árbol de derivación para la verificación (en este caso) de paréntesis balanceados, y probando cadenas válidas o no válidas para comprobar el diferente comportamiento del programa con diferentes cadenas.

PLANTEAMIENTO DEL PROBLEMA

El problema que se presenta es que por medio de un programa en cualquier lenguaje de programación, se comprueben (en este caso) que los parentesis ingresados por teclado para una expresión son correctos.

Para darle solución a este problema se optó por hacer este programa en lenguaje C, y por medio de una pila irá leyendo la cadena ingresada caracter a caracter para buscar un parentesis abierto y meterlo a la pila, hasta que encuentre un paréntesis cerrado y lo saque de la pila para compobar si los paentesis ingresados están alanceados.

DISEÑO DE LA SOLUCIÓN



Este es una representación de un árbol dónde se hará la derivación de la expresión ingresada para así poder verificar si los paréntesis están correctamente balanceados.

En el nodo raíz es dónde eventualmente partiremos la derivación.

Los nodos interiores harán referencia a el contenido de los parentesis y de cómo se irá desglosando la expresión

Mientras que en los nodos hoja se guardará la expresión final.

IMPLEMENTACIÓN DE LA SOLUCIÓN

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <ctype.h>

#define MAX 100
#define TRUE      1
#define FALSE     0

typedef unsigned char boolean;

typedef struct elemento{
    char c;
    float n;
}elemento;
typedef struct nodo{
    elemento e;
    struct nodo *abajo;
}nodo;
typedef struct pila{
    nodo *tope;
}pila;
void Inicializar(pila *s){
    s -> tope = NULL;
    return;
}
void Push(pila *s, elemento e){
    nodo *aux;
    aux = (nodo *)malloc(sizeof(nodo));
    (*aux).e = e;
    aux -> abajo = s -> tope;
    s -> tope = aux;
    return;
}
elemento Top(pila *s){
    return s -> tope -> e;
}
```

```

elemento Pop(pila *s){
    elemento r;
    nodo *aux;
    r = s -> tope -> e;
    aux = s -> tope;
    s -> tope = s -> tope -> abajo;
    free(aux);
    return r;
}

boolean Empty(pila *s){
    return (s -> tope == NULL)?TRUE:FALSE;
}

int Size(pila *s){
    int size = 0;
    nodo *aux;
    aux = s -> tope;
    while( aux != NULL){
        aux = aux -> abajo;
        size++;
    }
    return size;
}

void Destroy(pila *s){
    nodo *aux;
    while(s -> tope != NULL){
        aux = s -> tope -> abajo;
        free(s -> tope);
        s -> tope = aux;
    }
    return;
}

boolean otroProceso(){
    char sn, respuesta[20];
    printf("\n\n\t Desea introducir otra expresion ? \n escriba s para si y n
para no: ");
    fflush(stdin);
    scanf("%c",&sn);
    printf("\n Usted selecciono %c \n", sn);
    return (sn == 's' || sn == 'S')?TRUE:FALSE;
}

```

```

}
boolean validarParentesis2(char const *cadena){
    int i;
    elemento e1;
    pila p1;
    Inicializar(&p1);
    printf("\nElimina epsilon\n");
    for(i = 0; i < strlen(cadena); i++){
        if(cadena[i] == '('){

            e1.c = cadena[i];
            Push(&p1, e1);
            printf("( S");
        }
        else if(cadena[i] == '){

            if(Empty(&p1) == FALSE){
                Pop(&p1);
                printf(" ) ");
            }
            else{
                printf("\n\t%i ERROR, hay mas parentesis que
cierran que los que abren \n", i+1);
                return FALSE;
            }
        }
    }
    return (Empty(&p1) == TRUE)?TRUE:FALSE;
}

boolean validarParentesis(char const *cadena){
    int i;
    elemento e1;
    pila p1;
    Inicializar(&p1);
    printf("\n Su expresion fue guardada correctamente. \nSe procedera
a validar parentesis\n");
    for(i = 0; i < strlen(cadena); i++){
        if(cadena[i] == '('){

            e1.c = cadena[i];

```

```

        Push(&p1, e1);
        printf("( S");
    }
    else if(cadena[i] == '){
        if(Empty(&p1) == FALSE){
            Pop(&p1);
            printf(" ) eps ");
        }
        else{
            printf("\n\t%i ERROR, hay mas parentesis que
cierran que los que abren \n", i+1);
            return FALSE;
        }
    }
}
validarParentesis2(cadena);
return (Empty(&p1) == TRUE)?TRUE:FALSE;
}

```

```

int main(){
    elemento e1;
    pila p1;
    char expresion[MAX];
    int i;
    while(1){
        Inicializar(&p1);
        printf("Programa que valida y resuelve expresiones
algebraicas por medio de una pila\n");
        printf("Como ejemplo, puede introducir una expresion como ( )
\n");

        printf("NOTA: procure no dejar espacios\n");
        printf("\n\n A continuacion introduzca su expresion: ");
        scanf("%s",expresion);
        if(validarParentesis(expresion) == TRUE){
            printf("\nExpresionfinal\n%s", expresion);
            printf("\nParentesis correctos\n");
            Destroy(&p1);
        }
    }
}

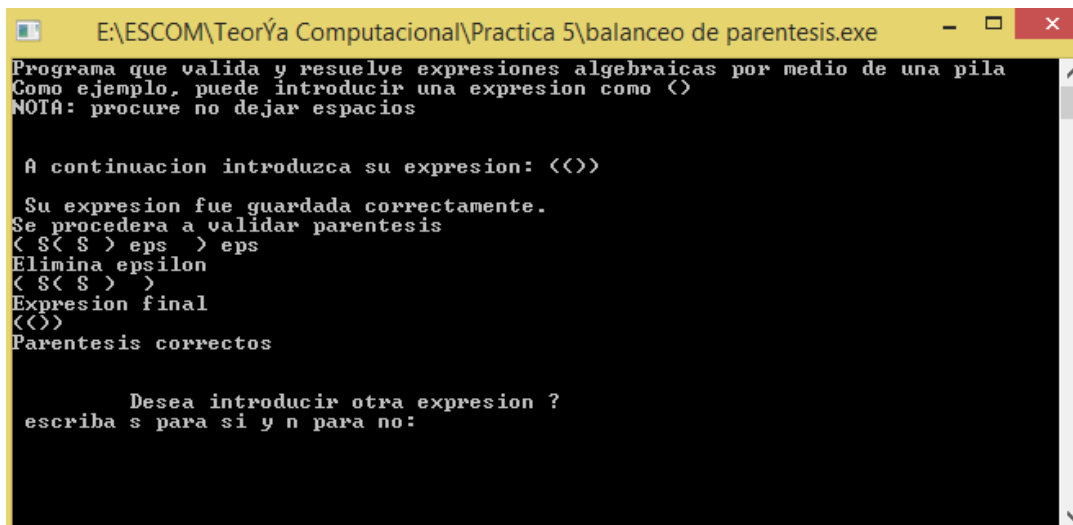
```

```

        else{
            printf("\n Parentesis incorrectos, revise la sintaxis \n");
        }
        if(otroProceso() == TRUE){
            Destroy(&p1);
            getchar();
            setbuf(stdin, NULL);
        }
        else break;
    }
    getchar();
    return 0;
}

```

FUNCIONAMIENTO



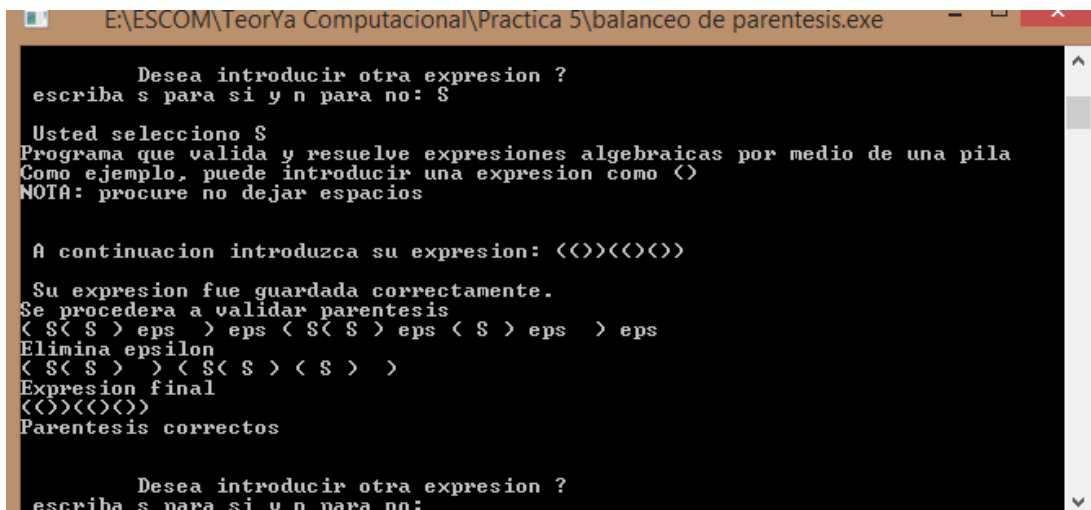
```

E:\ESCOM\TeorYa Computacional\Practica 5\balanceo de parentesis.exe
Programa que valida y resuelve expresiones algebraicas por medio de una pila
Como ejemplo, puede introducir una expresion como <>
NOTA: procure no dejar espacios

A continuacion introduzca su expresion: <>>
Su expresion fue guardada correctamente.
Se procedera a validar parentesis
< $< $ > eps > eps
Elimina epsilon
< $< $ > >
Expresion final
<<>>
Parentesis correctos

Desea introducir otra expresion ?
escriba s para si y n para no:

```



```

E:\ESCOM\TeorYa Computacional\Practica 5\balanceo de parentesis.exe

Desea introducir otra expresion ?
escriba s para si y n para no: s

Usted selecciono S
Programa que valida y resuelve expresiones algebraicas por medio de una pila
Como ejemplo, puede introducir una expresion como <>
NOTA: procure no dejar espacios

A continuacion introduzca su expresion: <>><><>
Su expresion fue guardada correctamente.
Se procedera a validar parentesis
< $< $ > eps > eps < $< $ > eps < $ > eps > eps
Elimina epsilon
< $< $ > > < $< $ > < $ > >
Expresion final
<<>><><>
Parentesis correctos

Desea introducir otra expresion ?
escriba s para si y n para no: _

```

```
E:\ESCOM\Teoría Computacional\Practica 5\balanceo de parentesis.exe

Usted selecciono s
Programa que valida y resuelve expresiones algebraicas por medio de una pila
Como ejemplo, puede introducir una expresion como <>
NOTA: procure no dejar espacios

A continuacion introduzca su expresion: <><

Su expresion fue guardada correctamente.
Se procedera a validar parentesis
< S< S > eps < S
Elimina epsilon
< S< S > < S
Parentesis incorrectos, revise la sintaxis

Desea introducir otra expresion ?
escriba s para si y n para no: n

Usted selecciono n

-----
Process exited after 165.6 seconds with return value 0
Presione una tecla para continuar . . .
```

CONCLUSIONES

En ésta práctica se realizaron programas para una gramática libre de contexto en dónde se validaron paréntesis por medio de una pila y se ingresaron cadenas para verificar si dicha expresión ingresada fue correcta o si la cadena que ingresó era no válida.

Al hacer el árbol de derivación, me di cuenta que es muy difícil representarlo en el lenguaje C, pero me di cuenta que es lo mismo que si imprimes en pantalla las derivaciones una por una, y al final imprimes el resultado final, que si se cumple la codición de que estén balanceados, será la expresión que se ingresó inicialmente.

BIBLIOGRAFÍA

1. <http://teodelacomp.blogspot.com/2011/03/arboles-de-derivacion.html>
2. http://webdiis.unizar.es/asignaturas/LGA/material_2003_2004/4_LIC_GIC_ADPND.pdf