



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

UNIVERSIDAD DE LA FUERZAS ARMADAS ESPE

ESTRUCTURA DE DATOS



Integrantes: Arévalo Dalton, Jiménez Diego, Rivera Stalin, Zapata Lizzette.

Docente: Ing. Fernando Solís

NRC: 3685

INFORME

TEMA: Generación de la calculadora polaca mediante la implementación de árboles.

1. OBJETIVO GENERAL

Generar un programa donde el usuario logre insertar una expresión matemática por medio del teclado siguiendo la lógica de la polaca mediante el uso de árboles y la implementación de interfaz gráfica del árbol creado.

2. OBJETIVO ESPECÍFICOS

- a. Documentar de manera correcta el código desarrollado mediante el uso de Cdocs lo cual servirá para la mejor comprensión del código.
- b. Realizar un informe detallado todo el proceso de documentación del proyecto resaltando partes importantes sobre su desarrollo.
- c. Aplicar correctamente el tema de árboles aprendidos en clase, sirviendo como un pilar fundamental de la estructura de datos.

3. DESARROLLO

3.1 PLANTEAMIENTO DEL PROBLEMA.

Se pide desarrollar un programa que permita ingresar una expresión matemática siguiendo la lógica de la inversa polaca y con dicha expresión crear un árbol, el cual mediante una interfaz gráfica imprime la imagen del árbol creado en un archivo .jpg o .png, además de la implementación de los respectivos cdocs y documentación

3.2 DESCRIPCIÓN DEL PROCESO.

3.2.2 POLACA INVERSA.

La polaca inversa es una forma distinta de escribir expresiones matemáticas enfocado principalmente en la introducción de datos, donde sigue la lógica de la calculadora polaca pero en sentido contrario donde están los operandos y después viene el operador que va a realizar los cálculos sobre ellos esto con el fin de para reducir el acceso de la memoria de computadora y para usar el stack para evaluar expresiones.

El objetivo principal del uso de la polaca inversa es evaluar los datos introducidos para ser manejados en una estructura de característica última en entrar, primero en salir. Caracterizado dado a que los cálculos se realizan secuencialmente, primero introduciendo sus operadores y después las expresiones por ellos resulta fácil de manejar.

3.2.3 USO DE ÁRBOLES.

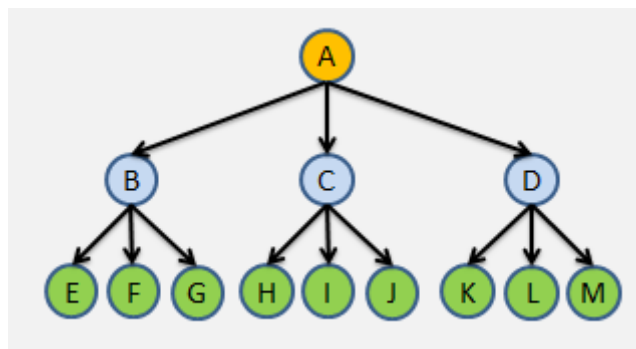
Los árboles son utilizados en la estructura de datos como una forma de organización almacenando sus nodos en una forma de jerarquía a diferencia de las listas, pilas o colas se debe tener en cuenta la manera de estructura de un árbol siendo así:

- Nodos: Se le llama Nodo a cada elemento que contiene un Árbol.
- Nodo Raíz: Se refiere al primer nodo de un Árbol, Solo un nodo del Árbol puede ser la Raíz.
- Nodo Padre: Se utiliza este término para llamar a todos aquellos nodos que tienen al menos un hijo.

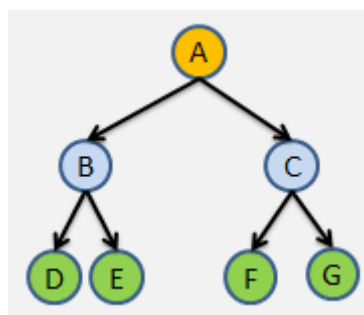
- **Nodo Hijo:** Los hijos son todos aquellos nodos que tiene un padre.
- **Nodo Hermano:** Los nodos hermanos son aquellos nodos que comparten a un mismo padre en común dentro de la estructura.
- **Nodo Hoja:** Son todos aquellos nodos que no tienen hijos, los cuales siempre se encuentran en los extremos de la estructura.
- **Nodo Rama:** Estos son todos aquellos nodos que no son la raíz y que además tiene al menos un hijo.

Tipos de árboles

- **Árbol n-ario:** El número máximo de hijos por nodo es de N



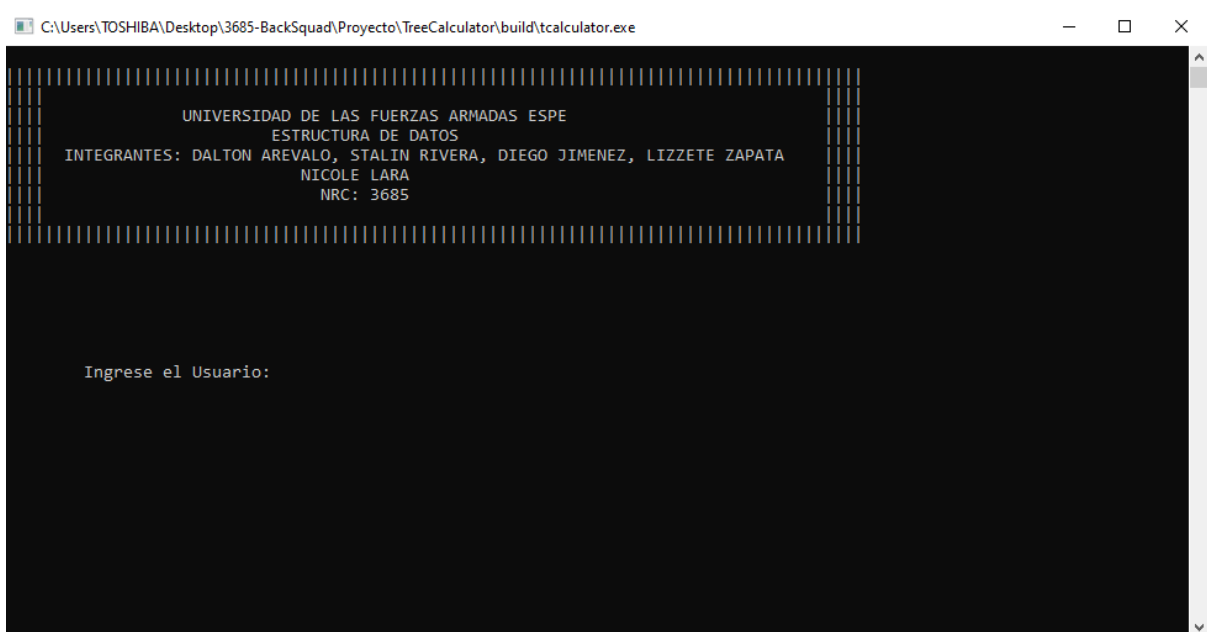
- **Árboles binarios:** Esta estructura donde cada nodo sólo puede tener máximo 2 hijos de Grado 2.



3.2.1 USO DE CLASES.

En el proceso de desarrollo del proyecto se aplicó el uso de clases para ello es importante tener en claro los conceptos de clases y objetos en c++ una clase es un nuevo tipo de dato que sirve para crear objetos en la programación y así se obtiene una consistencia lógica relacionada con sus miembros. (Aguilar, 2007)

3.3 CAPTURAS CORRIDA DEL PROGRAMA.



```
C:\Users\TOSHIBA\Desktop\3685-BackSquad\Proyecto\TreeCalculator\build\tcalculator.exe

UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
ESTRUCTURA DE DATOS
INTEGRANTES: DALTON AREVALO, STALIN RIVERA, DIEGO JIMENEZ, LIZZETE ZAPATA
NICOLE LARA
NRC: 3685

Ingrese el Usuario:
```

```
C:\Users\TOSHIBA\Desktop\3685-BackSquad\Proyecto\TreeCalculator\build\calculator.exe

UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
ESTRUCTURA DE DATOS
INTEGRANTES: DALTON AREVALO, STALIN RIVERA, DIEGO JIMENEZ, LIZZETE ZAPATA
NICOLE LARA
NRC: 3685

Ingrese el Usuario: BackSquad
Clave: *****

Sea Bienvenido

Presione una tecla para continuar . . .
```

```
C:\Users\TOSHIBA\Desktop\3685-BackSquad\Proyecto\TreeCalculator\build\calculator.exe

.4)Calculadora - AB
Mostrar Expresion
Imprimir arbol
Salir

La funcion esta bien ingresada
La funcion es correcta
Orden postfijo (Posorden): 45 32 2 2 ^ exp 3 3 3 * + cos * * 23.4 tan + +
Notacion prefija: + + tan 23.4 * * cos + * 3 3 3 exp ^ 2 2 32 45
Notacion prefija: 45+32*exp(2^2)*cos(3+3*3)+tan(23.4)
Notacion funcional: suma(45, suma(multiplicacion(32, multiplicacion(exp(potenciacion(2, 2)), cos(suma(3, multiplicacion(
3, 3))))), tan(23.4)))
Resultado: 1525.45

Presione una tecla para continuar . . .
```



```
C:\Users\TOSHIBA\Desktop\3685-BackSquad\Proyecto\TreeCalculator\build\tcalculator.exe
.4)Calculadora - AB
Mostrar Expresion
Imprimir arbol
Salir

Recorrido: 45 -> 32 -> 2 -> 2 -> 3 -> 3 -> 3 -> 23.4 ->

Presione una tecla para continuar . . .
```

3.4 UNIT TEST.

UnitTest1.cpp X

UnitTest > UnitTest1.cpp > ...

```
10
11 #include "pch.h"
12 #include "CppUnitTest.h"
13 #include "../TreeCalculator/tnode.h"
14
15 using namespace Microsoft::VisualStudio::CppUnitTestFramework;
16 using namespace std;
17
18 namespace UnitTest{
19     TEST_CLASS(UnitTest1){
20     public:
21         TEST_METHOD(TestMethod1){
22             TNode tnodo = new TNode(15);
23             Assert::IsNotNull(tnodo);
24         }
25     };
26
27     TEST_CLASS(UnitTest2){
28     public:
29         TEST_METHOD(TestMethod1){
30             TNode<std::string>* tmp = str.getCursor();
31             Assert::IsNotNull(tmp);
32         }
33     };
34 }
```

Live Share ⓘ CMake: [Debug]: Ready ⚙ [GCC 10.2.0 x86_64-w64-mingw32] ⚙ Build [all] ⓘ


```
10
11 #include "pch.h"
12 #include "CppUnitTest.h"
13 #include "../TreeCalculator/opstree.h"
14
15 using namespace Microsoft::VisualStudio::CppUnitTestFramework;
16 using namespace std;
17
18 namespace UnitTest{
19     TEST_CLASS(UnitTest1){
20     public:
21         TEST_METHOD(TestMethod1){
22             Node<std::string>* tmp = str.getCursor();
23             Assert::IsNotNull(tmp);
24         }
25     };
26
27     TEST_CLASS(UnitTest2){
28     public:
29         TEST_METHOD(TestMethod1){
30             TNode<std::string> tnode(tmp->getDat());
31             Assert::IsNotNull(tmp);
32         }
33     };
34 }
```

UnitTest > UnitTest3.cpp > ...

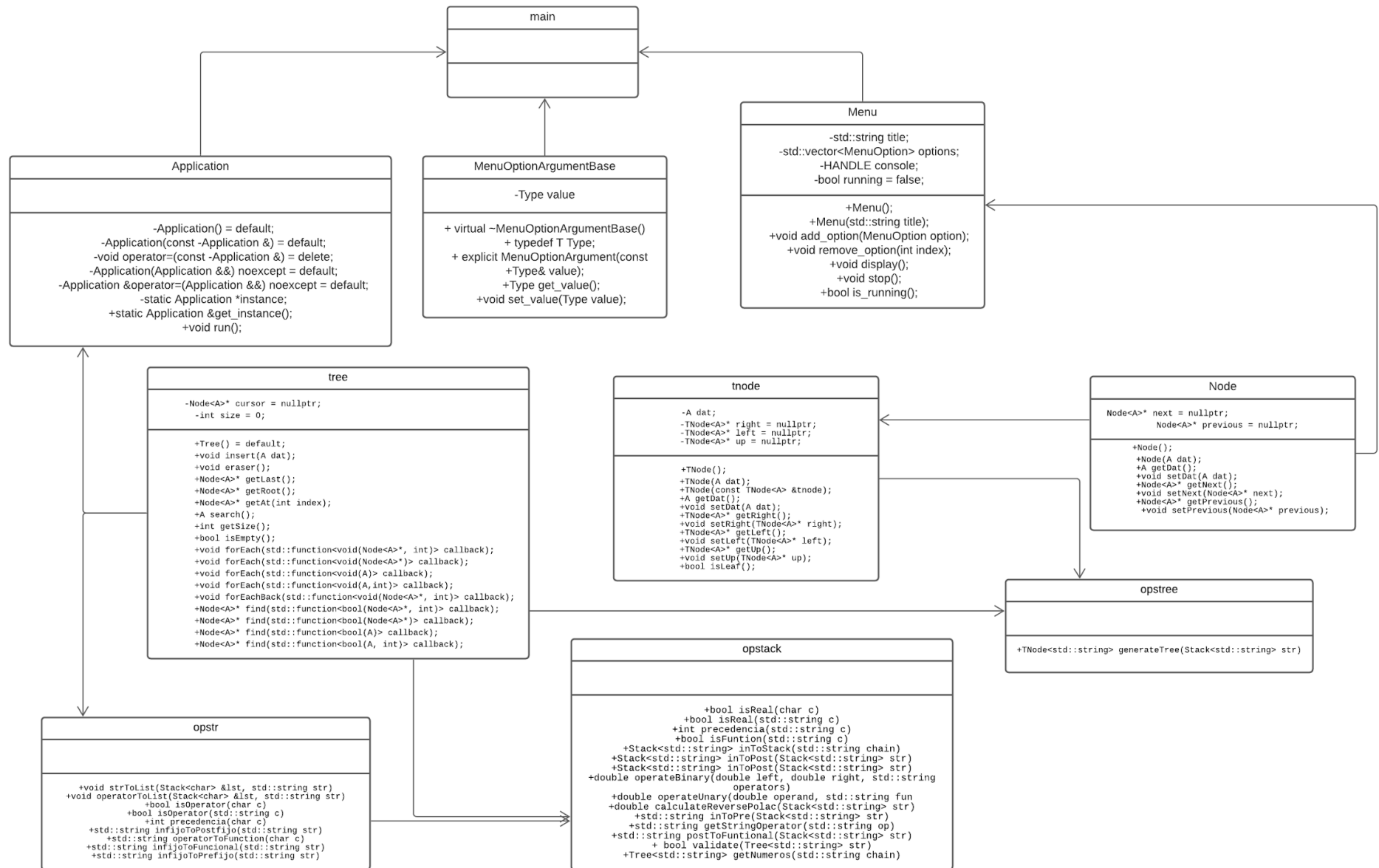
```
19     TEST_CLASS(UnitTest1){
20     public:
21         TEST_METHOD(TestMethod1){
22             void strToList(Tree<char> &lst, std::string str){
23                 char* cadena = new cadena('1', '0', '+', '5', '*', '8', '/', '2');
24                 Assert::IsNotNull(cadena);
25             }
26         }
27     };
28
29     TEST_CLASS(UnitTest2){
30     public:
31         TEST_METHOD(TestMethod1){
32             std::string infijoToPostfijo(std::string str){
33                 char* cadena = new cadena('s', 'i', 'n', '(', '8', ')', '/', '2');
34                 Assert::IsNotNull(cadena->infijoToFuncional("sin(8)/2"));
35             }
36         };
37     }
38
39     TEST_CLASS(UnitTest3){
40     public:
41         TEST_METHOD(TestMethod1){
42             std::string infijoToPrefijo(std::string str){
43                 char* cadena = new cadena('2');
44                 Assert::IsNotNull(cadena->infijoToFuncional("2"));
```

```

19     TEST_CLASS(UnitTest1){
20     public:
21         TEST_METHOD(TestMethod1){
22             Tree<std::string> inToTree(std::string chain){
23                 char* str = new('8', '/', '2');
24                 Assert::IsNotNull(str);
25             }
26         }
27     };
28
29     TEST_CLASS(UnitTest2){
30     public:
31         TEST_METHOD(TestMethod1){
32             Tree<std::string> inToPost(Tree<std::string> str){
33                 Node<std::string>* tmp = str.getCursor();
34                 Assert::IsNotNull(tmp));
35             }
36         };
37     }
38
39     TEST_CLASS(UnitTest3){
40     public:
41         TEST_METHOD(TestMethod1){
42             calculateReversePolac(Stack<std::string> str){
43                 Node<std::string>* tmp = str.getCursor();
44                 Assert::IsNotNull(isFuntion(tmp->getDat()));

```

3.5 DIAGRAMA DE CLASE



4. CONCLUSIONES.

Para concluir, con la realización del proyecto grupal se ha tomado en cuenta que el árbol es importante en la estructura de datos pues permite almacenar cantidades finitas de datos, todo esto en una forma ordenada en el computador.

Un árbol se compone de un conjunto de nodos que están enlazados por medio de ramas. donde es de gran importancia la raíz pues de aquí parte todo y de aquí viene su nombre de árbol, dada a su similitud. Partiendo de la raíz se puede llegar a cualquier nodo a través de sus ramas y distintos niveles sucesivos formando una especie de camino.

Tomar en cuenta los recorridos de los árboles entendiendo que mediante este proceso se puede acceder a los distintos nodos del árbol y esto va a hacer que varíen la cantidad de nodos existentes en dicho árbol.

5. RECOMENDACIONES

Emplear de manera correcta los términos aprendidos de lo contrario existirá confusión al momento del desarrollo del proyecto, culminando en errores que perjudicaron a todo el equipo de trabajo en la calificación final.

Se debe saber dónde estamos ubicados en el árbol, y cuando se vaya haciendo más extenso el algoritmo o sea más fácil de ubicarnos. Ya que de lo contrario se creará un mal funcionamiento del algoritmo, borrar información equivocada o que ocurra un desbordamiento de datos.

6. BIBLIOGRAFÍA

González, A. (2002). Definiciones: conjuntos, grafos, y árboles

González, A. H., & Gallo, S. L. (2021). Árboles binarios ordenados.

Joyanes, L. A. (2007). *Estructura de datos c++*. Madrid: McGraw - Hill / Interamericana.

Jorge, S. Estructuras de datos: Arboles binarios de busqueda, Monticulos.

Kruse, R. L., & Miguel, E. A. (1988). Estructura de datos y diseño de programas (No. 005.1/K94dE). Prentice-Hall Hispanoamericana.

Loomis, M. E. (1991). Estructura de datos y organización de archivos. Prentice-Hall Hispanoamericana.