MANUAL DE USUARIO

Proyecto No. 2
LABORATORIO ORGANIZACION DE LENGUAJES Y COMPILADORES 1 Sección C

José Eduardo Galdámez González
Carne: 202109732

Manual de Usuario:

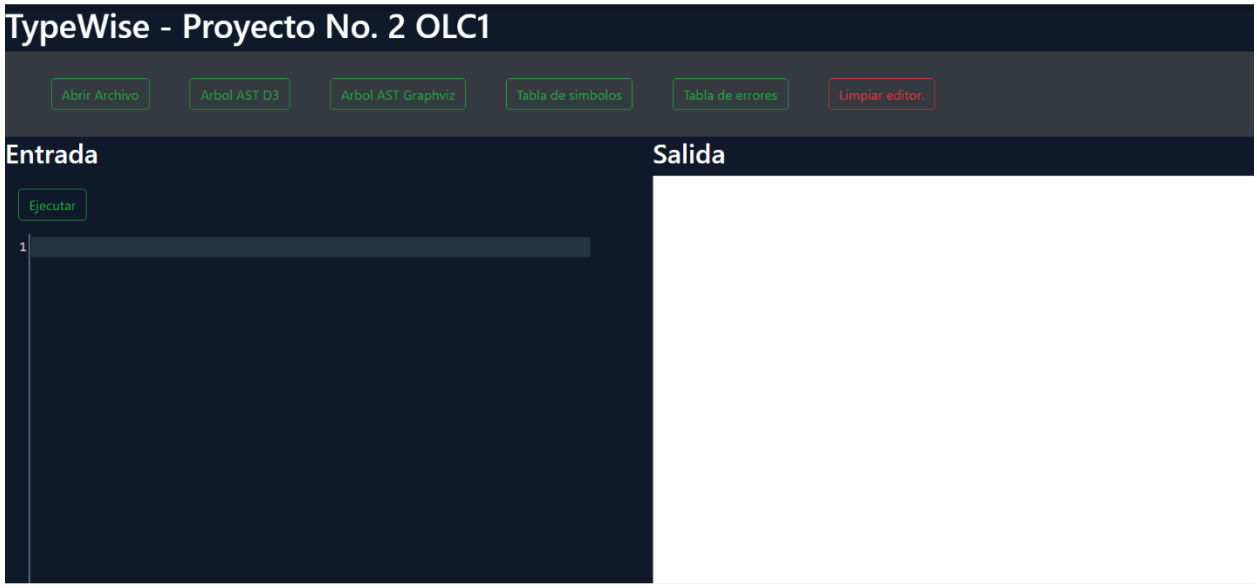
El manual de usuario que se presenta tiene como objetivo guiar a los usuarios en el uso del lenguaje de programación TypeWise, el cual fue desarrollado como un intérprete para los estudiantes de Introducción a la Programación y Computación 1 de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

Este manual proporcionará a los usuarios una visión general del programa, así como las instrucciones detalladas sobre cómo utilizar el lenguaje de programación TypeWise para crear, guardar y ejecutar programas. Además, se describirán las funciones principales del lenguaje de programación, así como las características y herramientas que estarán disponibles para los usuarios.

Para hacer uso de TypeWise, es necesario tener conocimientos previos de programación y haber seguido los conceptos básicos de la asignatura. Este manual explicará cómo ingresar el código fuente, cómo interpretar y ejecutar programas, y cómo utilizar las funciones principales del lenguaje de programación.

En resumen, este manual de usuario tiene como objetivo facilitar la comprensión del lenguaje de programación TypeWise y permitir a los usuarios aprovechar al máximo sus funcionalidades para el aprendizaje y desarrollo de habilidades en programación.

Menú Principal:



Funcionalidades

- Abrir archivos: El editor deberá abrir archivos .tw
- Ejecutar: hará el llamado al intérprete, el cual se hará cargo de realizar los análisis
- Limpiar editor: Ayudara a limpiar toda el área de editor.

Reportes

- Reporte de Errores: Se mostrarán todos los errores encontrados al realizar el análisis léxico, sintáctico.
- Generar Árbol AST (Árbol de Análisis Sintáctico): se debe generar una imagen del árbol de análisis sintáctico que se genera al realizar los análisis.
- Reporte de Tabla de Símbolos: Se mostrarán todas las variables, métodos y funciones que han sido declarados dentro del flujo del programa.
-

MANEJO DE LENGUAJE:

Comentarios: Los comentarios son una forma elegante de indicar que función tiene cierta sección del código que se ha escrito simplemente para dejar algún mensaje en específico.

```
// Este es un comentario de una línea

/*
        Este es un comentario
         Multilínea
         Para este lenguaje
*/
```

Tipos de Datos

Los tipos de dato que soportará el lenguaje en concepto de un tipo de variable se definen a continuación:

TIPO	DEFINICION	DESCRIPCION	EJEMPLO	OBSERVACIONES	DEFAULT
Entero	Int	Este tipo de datos aceptará solamente números enteros.	1, 50, 100, 25552, etc.	Del -2147483648 al 2147483647	0
Doble	Double	Admite valores numéricos con decimales.	1.2, 50.23, 00.34, etc.	Se manejará cualquier cantidad de decimales	0.0
Booleano	Boolean	Admite valores que indican verdadero o falso.	True, false	Si se asigna un valor booleano a un entero se tomará como 1 o 0 respectivamente.	True
Caracter	Char	Tipo de dato que únicamente aceptará un único carácter, y estará delimitado por comillas simples. ' '	'a', 'b', 'c', 'E', 'Z', '1', '2', '^', '%', ')', '=', '!', '&', '/', '\\', '\n', etc.	En el caso de querer escribir comilla simple escribir se escribirá \ y después comilla simple \', si se quiere escribir \ se escribirá dos veces \\, existirá también \n, \t, \r, \".	'\u0000' (carácter 0)
Cadena	String	Es un grupo o conjunto de caracteres que pueden tener cualquier carácter, y este se encontrará delimitado por comillas dobles. " "	"cadena1", "-- ** cadena 1"	Se permitirá cualquier carácter entre las comillas dobles, incluyendo las secuencias de escape: \" comilla doble \\ barra invertida \n salto de linea \r retorno de carro \t tabulación	"" (string vacío)

```
1 int a;
2 String b;
3 double c;
```

Secuencias de Escape

Las secuencias de escape se utilizan para definir ciertos caracteres especiales dentro de cadenas de texto. Las secuencias de escape disponibles son las siguientes:

SECUENCIA	DESCRIPCION	EJEMPLO
\n	Salto de línea	"Hola\nMundo"
\\	Barra invertida	"C:\\miCarpeta\\Personal"
\"	Comilla doble	"\"Esto es una cadena\""
\t	Tabulación	"\tEsto es una tabulación"
\'	Comilla Simple	"\'Estas son comillas simples\'"

Operadores Aritméticos:

Se manejan las operaciones básicas que son:

- Suma
- Resta
- Multiplicación
- División
- Potencia
- Módulo
- Negación Unaria

Operadores relaciones:

OPERADOR	DESCRIPCIÓN	EJEMPLO
==	Igualación: Compara ambos valores y verifica si son iguales: - Iguales= True - No iguales= False	1 == 1 "hola" == "hola" 25.5933 == 90.8883 25.5 == 20
!=	Diferenciación: Compara ambos lados y verifica si son distintos. - Iguales= False - No iguales= True	1 != 2, var1 != var2 25.5 != 20 50 != 'F' "hola" != "hola"
<	Menor que: Compara ambos lados y verifica si el derecho es mayor que el izquierdo. - Derecho mayor= True - Izquierdo mayor= False	(5/(5+5))<(8*8) 25.5 < 20 25.5 < 20 50 < 'F'
<=	Menor o igual que: Compara ambos lados y verifica si el derecho es mayor o igual que el izquierdo. - Derecho mayor o igual= True - Izquierdo mayor= False	55+66<=44 25.5 <= 20 25.5 <= 20 50 <= 'F'
>	Mayor que: Compara ambos lados y verifica si el izquierdo es mayor que el derecho. - Derecho mayor= False - Izquierdo mayor= True	(5+5.5)>8.98 25.5 > 20 25.5 > 20 50 > 'F'
>=	Mayor o igual que: Compara ambos lados y verifica si el izquierdo es mayor o igual que el derecho. - Derecho menor o igual= True - Izquierdo menor= False	5-6>=4+6 25.5 >= 20 25.5 >= 20 50 >= 'F'

Operadores Lógicos:

OPERADOR	DESCRIPCIÓN	EJEMPLO	OBSERVACIONES
||	**OR:** Compara expresiones lógicas y si al menos una es verdadera entonces devuelve verdadero en otro caso retorna falso	(55.5)|| bandera==true Devuelve true	bandera es true
&&	**AND**: Compara expresiones lógicas y si son ambas verdaderas entonces devuelve verdadero en otro caso retorna falso	(flag1) && ("hola" == "hola") Devuelve true	flag1 es true
!	**NOT:** Devuelve el valor inverso de una expresión lógica si esta es verdadera entonces devolverá falso, de lo contrario retorna verdadero.	!var1 Devuelve falso	var1 es true

Caracteres de finalización y encapsulamiento de sentencias

El lenguaje se verá restringido por dos reglas que ayudan a finalizar una

instrucción y encapsular sentencias:

- Finalización de instrucciones: para finalizar una instrucciones se utilizara el signo ; .
- Encapsular sentencias: para encapsular sentencias dadas por los ciclos, métodos, funciones, etc, se utilizará los signos { y }.

```
//Ejemplo de finalización de instrucción
int edad = 18;
//Ejemplo de encapsulamiento de sentencias
If(i==1){
    int a=15;
    Print("soy el numero "+a);
}
```

Declaración y asignación de variables

Una variable deberá de ser declarada antes de poder ser utilizada. Todas las variables tendrán un tipo de dato y un nombre de identificador. Las variables podrán ser declaradas global y localmente.

```
string cadena = "hola";
char var_1 = 'a';
boolean verdadero;
```

Casteos: Proceso de convertir un tipo de datos en otro. El casting es útil cuando se necesita convertir un valor de un tipo a otro para poder realizar ciertas operaciones o para asegurarse de que los valores sean compatibles.

```
1 int edad = (int) 18.6; //toma
2 char letra = (char) 70; //tom
3 double numero = (double) 16;
```

Estructuras de datos o Vectores:

Los vectores son una estructura de datos de tamaño fijo que pueden almacenar valores de forma limitada, y los valores que pueden almacenar son de un único tipo; int, double, boolean, char o string.

El lenguaje permitirá únicamente el uso de arreglos de una dimensión.

- Declaración de vectores
- Acceso a vectores
- Modificación de vectores

```
1  //declaracion 1
2  String[] vector2 = {"hola", "Mundo"};
3
4  //declaracion 1
5  int[] vectorNumero = {2020,2021,2022};
6
7  //acceso
8  vector2[0] = "OLC1";
9
10 //modificacion
11 vector2[1] = "1er Semestre " + vectorNumero[1];
```

Sentencias cíclicas

For: El ciclo o bucle for, es una sentencia que nos permite ejecutar N cantidad de veces la secuencia de instrucciones que se encuentra dentro de ella.

While: El ciclo o bucle While, es una sentencia que ejecuta una secuencia de instrucciones mientras la condición de ejecución se mantenga verdadera.

Do While: El ciclo o bucle Do-While, es una sentencia que ejecuta al menos una vez el conjunto de instrucciones que se encuentran dentro de ella y que se sigue ejecutando mientras la condición sea verdadera.

```
int x = 0;

while(x<5){
  print(x);
  x++;
}
```

```
int a=5;
do{
    if (a>=1 && a <3){
    print(true);
    }else{
      print(false);
    }
    a--;
} while (a>0);
```

```
for(int i = 0; i<10; i++){
  print("x");
}
```

Funciones

Puede declarar funciones y métodos para retornar datos de cualquier tipo permitido por el lenguaje. A continuación, se muestran algunos ejemplos de la gramática de las mismas:

```
int factorialIterativo(int n){
    int resultado = 1;
    for (int i = 1; i <= n; i++) {
        resultado = resultado * i;
    }
    return resultado;
}

int factorialRecursivo(int n) {
    if (n == 0) {
        return 1;
    }
    return (n * factorialRecursivo(n - 1));
}
```

```
//Ejemplo 1
void funcion1(){
print("hola");
 }

main funcion1();
/*RESULTADO
      hola
*/

//Ejemplo 2
void funcion2(string mensaje){
      print(mensaje);
}

main funcion2("hola soy un mensaje");
/*RESULTADO
      Hola soy un mensaje
*/
```

```
void holamundo(){
      print("Hola mundo");
}
```