Manual de Usuario

Organización de Lenguajes y Compiladores 1 – Proyecto 2

Aldo Saúl Vásquez Moreira
202109754

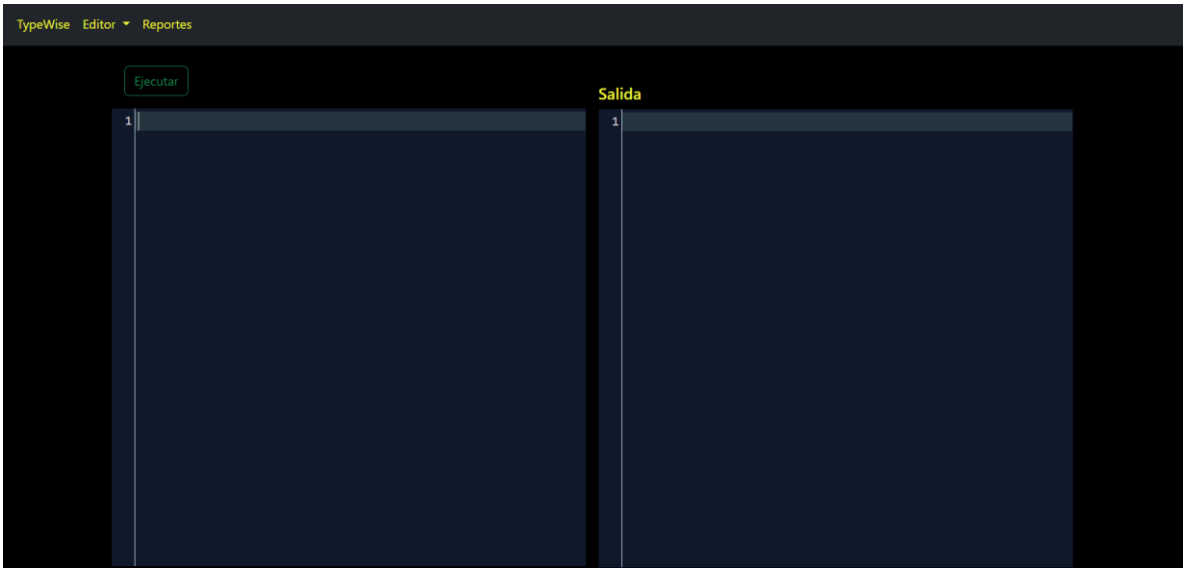
TYPEWISE

DESCRIPCIÓN DEL OBJETIVO DEL SOFTWARE DESARROLLADO

El curso de Organización de Lenguajes y Compiladores 1, perteneciente a la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala necesita un intérprete para que los estudiantes de Introducción a la Programación y Computación 1 utilicen para sus primeras prácticas. Por ello, se buscó desarrollar un lenguaje con sintaxis sencilla para facilitar el entendimiento de la programación en los estudiantes del curso.

FUNCIONAMIENTO BÁSICO DEL PROGRAMA

Para introducir al estudiante lo más rápidamente posible en el uso de este software se creó este manual. Comenzaremos a familiarizarlo a usted como usuario con la interfaz principal del software.



Como se puede observar es una interfaz muy sencilla y entendible. Cuenta con un submenú editor el cual contiene los siguientes submenús:

- **Crear archivo**

  El editor de texto tiene la capacidad de crear archivos nuevos en blanco, lo que significa que los usuarios pueden comenzar a escribir en un documento vacío desde cero. Para hacer esto, los usuarios pueden seleccionar "Archivo" y luego "Nuevo" o simplemente hacer clic en un botón específico para crear un nuevo archivo.

- **Abrir archivo**

  El editor de texto también tiene la capacidad de abrir archivos existentes, en este caso, archivos con extensión ".tw". Los usuarios pueden seleccionar "Archivo" y luego "Abrir" para buscar y abrir archivos con esta extensión específica. Una vez abierto el archivo, el usuario podrá ver y editar el contenido del archivo en la interfaz del editor de texto.

- **Guardar archivo**

  El editor de texto permite a los usuarios guardar el estado actual del archivo en el que están trabajando para evitar perder su progreso o cambios. Esto se puede hacer seleccionando "Archivo" y luego "Guardar" o "Guardar como". Si se selecciona "Guardar", se sobrescribirá la versión anterior del archivo con los cambios realizados. Si se selecciona "Guardar como", se pedirá al usuario que seleccione una ubicación y un nombre de archivo para el nuevo archivo.

- **Eliminar pestaña**

  El editor de texto tiene la capacidad de cerrar pestañas específicas si el usuario no desea trabajar con ellas. Esto se puede hacer haciendo clic en el botón "Cerrar pestaña" o seleccionando "Cerrar" en la pestaña específica que desea cerrar. Cuando se cierra una pestaña, se eliminará de la interfaz del editor de texto y se perderán todos los cambios que no se hayan guardado en ese archivo.

Asimismo, se cuenta con un área de reportes.

- **Reporte de errores**
  El editor de código tiene la capacidad de realizar análisis léxico y sintáctico del código escrito y mostrar los errores encontrados. Los errores pueden incluir problemas con la sintaxis, variables no declaradas, métodos y funciones inexistentes, entre otros. Cuando se encuentra un error, el editor lo resalta y muestra un mensaje de error en la parte inferior o superior de la interfaz. Los usuarios pueden hacer clic en el mensaje de error para obtener más detalles sobre el error y corregirlo.

- **Generar Árbol AST (Árbol de Análisis Sintáctico)**

  El editor de código tiene la capacidad de generar un árbol de análisis sintáctico (AST) a partir del código escrito por el usuario. El AST es una representación gráfica de la estructura sintáctica del código y ayuda a los usuarios a visualizar cómo se organizan las declaraciones y expresiones en el código. El AST generado puede mostrarse en una imagen y puede incluir detalles como los nodos y las ramas del árbol.

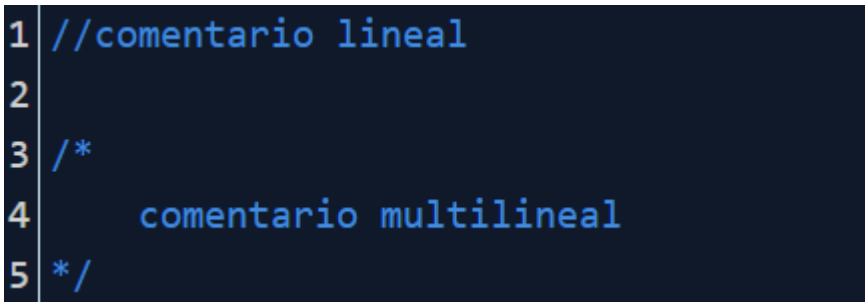
- **Reporte de Tabla de Símbolos**

  El editor de código también tiene la capacidad de mostrar una tabla de símbolos que muestra todas las variables, métodos y funciones declarados dentro del flujo del programa. Esta tabla es útil para ayudar a los usuarios a seguir el flujo de sus programas y para verificar la existencia de variables y métodos en diferentes partes del código. La tabla puede mostrarse en la interfaz del editor de código y puede incluir detalles como el tipo de variable, el alcance, el valor actual y la ubicación en el código donde se declaró.

LENGUAJE

Ya que se conoce el funcionamiento del software es momento de introducirlo al lenguaje que se maneja.

- **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. El lenguaje deberá soportar dos tipos de comentarios que son los siguientes.**
  - **Comentarios de una línea**
  - **Comentarios multilíneales**



- **Tipos de datos:**

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', '^', '%', ')', '=', 'ǀ', '&', '/', '\\', '\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)

A continuación, se muestra un ejemplo de su uso:

```
1 int numero = 10;          //Entero
2 double numero2 = 10.50;   //Doble
3 boolean valor = true;     //Booleano
4 char vocal = 'a';         //Caracter
5 String cadena = "hola";   //Cadena
6
```

- **Operaciones Aritméticas**

  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'

- **Operador ternario**

  El operador ternario es un operador que hace uso de 3 operandos para simplificar la instrucción 'if' por lo que a menudo este operador se le considera como un atajo para la instrucción 'if'. El primer operando del operador ternario corresponde a la condición que debe de cumplir una expresión para que el operador retorna como valor el resultado de la expresión segundo operando del operador y en caso de no cumplir con la expresión el operador debe de retornar el valor de la expresión del tercer operando del operador.

```
1 int edad = 18;
2 boolean banderaedad = false;
3 banderaedad = edad > 17 ? true : false;
```

- **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

- **Declaración de variables: La declaración de variables es una instrucción en un lenguaje de programación que reserva un espacio en la memoria del equipo para almacenar un valor. La declaración de variables es importante porque permite que un programa almacene y manipule datos. En la declaración de variables, se especifica el nombre de la variable y el tipo de dato que va a almacenar. El tipo de dato define el rango de valores que la variable puede contener y las operaciones que se pueden realizar con ella.**

```
1 String cadena = "hola";
2 char var_1 = 'a';
3 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 el valor entero de 18
2 char letra = (char) 70; //tomar el valor 'F' ya que el 70 en asc
3 double numero = (double) 16; //toma el valor 16.0
```

- **Incremento y decremento de variables: aumenta el valor en uno de la variable indicada.**

```
1 int x = 5;
2 x++; //incremento
3 x--; //decremento
```

- **Estructuras de datos**
    - **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];
12
```

- **Listas: Las listas son una estructura de datos que pueden crecer de forma iterativa y pueden almacenar hasta N elementos de un solo tipo; int, double, boolean, char o string.**
  - **Declaración de lista**
  - **Agregar valor a lista**
  - **Acceso a lista**
  - **Modificar lista**

```
list<String> listaS = new list<String>; //lista de strings
int[] vectorNumero = {2020,2021,2022};

//agregamos valores a la lista
listaS.add("Hola");
listaS.add("Mundo");

//antes de modificar
print(listaS[[0]]);
print(listaS[[1]]);

listaS[[0]]="OLC1";
listaS[[1]]="1er Semestre " + vectorNumero[1];

//despues de modificar
print("**********");
print(listaS[[0]]);
print(listaS[[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.**

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

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

```
int x = 0;

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

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

- **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));
}
```

- **Funciones especiales**
    - **Print: permite imprimir en pantalla lo que se desea.**
    - **toUpper: convierte una cadena en mayúsculas.**
    - **toLower: converte una cadena en minúsculas.**
    - **length: retorna la longitud de una cadena.**
    - **Truncate: redondea los números y retorna un entero.**
    - **typeOf: retorna el tipo de una variable.**
    - **toString: convierte un dato en cadena.**
    - **toCharArray: convierte una cadena en una lista de caracteres.**

Luego de esta guía, usted como usuario podrá usar el lenguaje y el software de la mejor manera para realizar las diversas pruebas de sus prácticas. Esperamos que sea de utilidad.