> ### Universidad de San Carlos de Guatemala
>
> #### Facultad de ingeniería
>
> #### Ingeniería en Ciencias y Sistemas

MANUAL DE USUARIO 📑

> **Elian Saúl Estrada Urbina**
>
> **201806838**

Acerca de JPR Editor ℹ️

JPR es un intérprete que ejecuta instrucciones de alto nivel definidas en el nuevo lenguaje exclusivo para los nuevos estudiantes de la Universidad de Sancarlos de Guatemala (USAC), el cuál es llamado JPR.

Requerimientos del sistema 💻

Sistema Operativo	Arquitectura	Versión	Memoria RAM
Windows	64 bits	Windows7 SP3 o superior	mínimo 2GB de memoria RAM
Linux	64 bits	Distribucines Debian 9 o superior	mínimo 2GB de memoria RAM

Instalación / Ejecución ▶️

Requisitos Opcionales

- Tener instalado `git`

Requisitos

- Tener instalado `python` en su versión 3.

1. Descargar los binarios

- Clonando el repositorio

```
git clone  https://github.com/ElianEstrada/OLC1_Project_201806838.git
```

- Descargar el archivo zip JPR Editor - Download

2. Dirigrise a la carperta de descarga.
3. Ejecutar la aplicación

```
phyton3 main.py
```

Interfaz 🔲

Esta es la vista principal que se mostrara al iniciar **JPR Editor**, esta divido en diferentes componentes que a continuación serán descritas:

Editor

ENTRADA

```
#*
   DECLARAMOS UN VECTOR DE 15 POSICIONES
   SE IMPRIMRÁ Y POSTERIORMENTE SE ORDENARÁ
*#
int[] vectorNumeros = new int[15];

var indiceFrase = 0;
string[] frase = new string[25]

func Hanoi(int discos, int origen, int auxiliar, int destino
    if (discos == 1) {
        print("Mover disco de " + origen + " a " + destino);
    } else {
        Hanoi(discos - 1, origen, destino, auxiliar);
        print("Mover disco de " + origen + " a " + destino);
        Hanoi(discos - 1, auxiliar, origen, destino);
    }
}

func imprimirVector(int[] miVector){
    for (var i = 0; i < length(miVector); i++) {
        print("vectorNumeros[" + i + "] = " + miVector[i]);
    }
}

func BubbleSort(int[] miVector){
    for (var i = 0; i < length(miVector); i++)
    {
        for (var j = 0; j < length(miVector) - i - 1; j++)
        {
```

La función principal del editor será el ingreso del código fuente que será analizado.

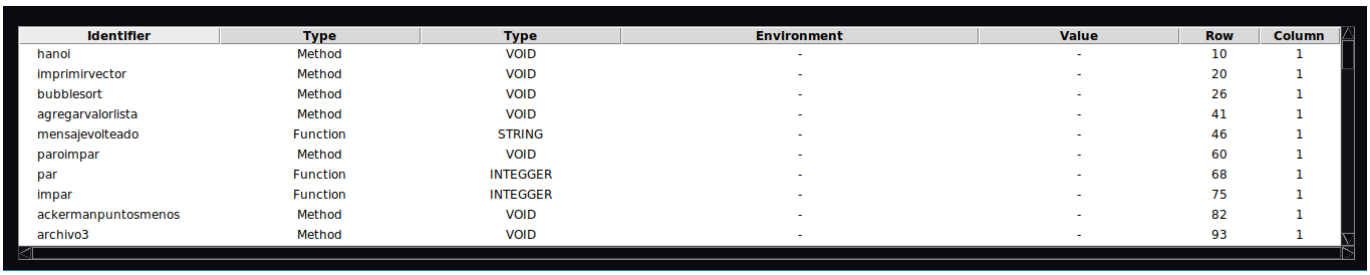
Principales Características

- Se podrán abrir diferentes archivos al mismo tiempo
- Cuenta con resaltado de sintaxis, para el lenguaje JPR.
- Barra lateral derecha con el número de línea.

Consola

La función principal de la consola será mostrar los resultados, mensajes de consola, mensajes de error del archivo entrada fue analizado.

Tabla de Símbolo



Identifier	Type	Type	Environment	Value	Row	Column
hanoi	Method	VOID	-	-	10	1
imprimirvector	Method	VOID	-	-	20	1
bubblesort	Method	VOID	-	-	26	1
agregarvalorlista	Method	VOID	-	-	41	1
mensajevolteado	Function	STRING	-	-	46	1
paroimpar	Method	VOID	-	-	60	1
par	Function	INTEGGER	-	-	68	1
impar	Function	INTEGGER	-	-	75	1
ackermanpuntosmenos	Method	VOID	-	-	82	1
archivo3	Method	VOID	-	-	93	1

La función principal de este componente es mostrar todas las variables, métodos y funciones que han sido declarados dentro del flujo del programa.

Barra lateral de Opciones

Cada opción realiza una funcionalidad distinta, que serán descritas a continuación de arriba hacia abajo:

Opción	Decripción
Abrir Archivo	Permite abrir un archvio con extensión *.jpr* guardado en nuestra máquina y muestra el contenido en el editor
Guardar Archivo	Permite guardar de forma local el contenido del editor.
Guardar Como..	Permite guardar de forma local el contenido del editor, permintiendo cambiar el nombre y ruta de archivo.
Ejecutar	Hará el llamado al intérprete, el cual se hará cargo de realizar los análisis léxico, sintáctico y semántico, además de ejecutar todas las sentencias.
Debugger	Característica que nos ayudará a ver el flujo de nuestro código al momento de ser ejecutado.
Reporte de Errores	Se mostrarán todos los errores encontrados al realizar el análisis léxico, sintáctico y semántico, este reporte se visualizará en un en navegador web.

Barra de estado

La barra de estado se encuentra en la parte inferior de la apicación Proporciona información sobre el documento en el que estás trabajando:

- Fila actual del cursor
- Columna actual del cursor
- Cantidad de errores encontrados luego de realizar el análisis de un archivo de entrada.

Sintaxis 📝

Case Insensitive

El lenguaje no distinguirá entre mayúsculas o minúsculas.

```
var a=0;
```

es lo mismo a decir

```
var A=0
```

Por lo tanto marcará error la declaración de `A` ya que la variable `a` ya existe previamente

Comentarios

- Comentarios de una línea

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

- Comentarios muliti línea

```
#*
    Este es un comentario
    Multilínea
    Para este lenguaje
    JPR!!!
    #$#”%$#”$%$#
*#
```

Caracteres de finalización y encapsulamiento de sentencias

- **Finalización de instrucciones:** para finalizar una instrucciones puede o no venir el signo punto y coma `;`.

```
var edad = 10;
var edad2 = 15
```

- **Encapsular sentencias:** para encapsular sentencias dadas por los ciclos, métodos, funciones, etc, se utilizará los signos `{` y `}`.

```
if(i == 1){
    var nota = 100;
    print(“Mi nota es de  ” + nota )
}
```

Declaración y asignación de variables

```
var numero; # null
var cadena = “hola” #String
var var_1 = ‘a’; # char
var verdadero;  # null
```

Casteos

El lenguaje aceptará los siguientes casteos:

- Int a double
- Double a Int
- Int a String
- Int a Char
- Double a String
- Char a int
- Char a double

```
# ‘(’<TIPO>’)’ <EXPRESION>

var edad = (int) 18.6;  #toma el valor entero de 18

var letra = (char) 70  #tomar el valor ‘F’ ya que el 70 en ascii es F

var numero = (double) 16; #toma el valor 16.0
```

Incremento y Decremento

```
var nota = 100;

# Incremento <EXPRESION>’+’’+’
nota++; #tiene el valor de 101

# Decremento <EXPRESION> ‘-‘‘-’
nota--; #tiene el valor 99
```

Arreglos

Declaración de Arreglos

```
# DECLARACION TIPO 1
# <TIPO> ‘[’ ‘]’ <ID> = new <TIPO> ( ’[’ <EXPRESION> ’]’ )+

int[ ] arr1 = new int[4]; #se crea un arreglo de 4 posiciones, con null en cada posición

int[ ] matriz1 = new int[2][2]; #se crea un arreglo de 4 posiciones, con null en cadaposición



# DECLARACION TIPO 2
# <TIPO> ‘[’ ‘]’ <ID> = ‘{’ <LISTAVALORES> ‘}’


string[ ] arreglo2 = {"hola", "Mundo"}; #arreglo de 2 posiciones, con "Hola" y "Mundo"

int[ ][ ] matriz = { {11, 12} , {21,22} }; #arreglo de 2 dimensiones, con 2 posiciones en cada
dimensión.
```

Acceso a Arreglos

```
# <ID> ( ’[’ EXPRESION ’]’ )+

string[ ] arr2 = {"hola", "Mundo"}; #creamos un arreglo de 2 posiciones de tipo string

string valorPosicion = arr2 [0]  #posición 0, valorPosicion = “hola”
```

Modificación de Arreglos

```
# <ID> ( ‘[’ EXPRESION ‘]’ )+ = EXPRESION

string[] arr2 = {"hola", "Mundo"}; #arreglo de 2 posiciones, con "Hola" y "Mundo"
int[] arrNumero = {2020,2021,2022};

arr2 [0] = “OLC1”;
arr2 [1] = “1er Semestre ”+ arrNumero [1];
```

Sentencias de control

if

```
if (x <50) {
    Print(“Menor que 50”);
    #Más sentencias
}
```

if-else

```
if (x <50) {
    print(“Menor que 50”);
    //Más sentencias
}
else {
    print(“Mayor que 100”);
    //Más sentencias
}
```

else-if

```
if (x <50) {
    print(“Menor que 50”);
    //Más sentencias
}
else if (x <= 50 && x > 0) {
    print (“Menor que 50”);
    //Más sentencias
}

else {
    print(“Número negativo”);
    //Más sentencias
}
```

Switch Case

```
var edad = 18;
switch( edad ) {
    case 10:
    Print(“Tengo 10 anios.”);
    # mas sentencias
    break;

    case 18:
    Print(“Tengo 18 anios.”);
    # mas sentencias

    case “25”:
    Print(“Tengo 25 anios en String.”);
    # mas sentencias
    break;

    default:
    Print(“No se que edad tengo. :(”);
    # mas sentencias
    break;
}
```

Sentencias cíclicas

While

```
while ( x < 100 ) {
    if ( x > 50 ) {
        print(“Mayor que 50”);
        #Más sentencias
    }
    else {
        print(“Menor que 100”);
        #Más sentencias
    }
    X++;
    #Más sentencias
}
```

For

```
for ( var i=0; i<3;i++ ) {
    print(“i=”+i)
    #más sentencias
}
```

Sentencias de transferencia

Break

```
for(var i = 0; i < 9; i++){
    if(i==5){
        print(“Me salgo del ciclo en el numero “ + i);
        break;
    }
    print(i);
}
```

Continue

```
for(var i = 0; i < 9; i++){
    if(i==5){
        print(“Me salte el numero ” + i);
        continue;
    }
    print(i);
}
```

Return

```
func sumar(int n1, int n2){
    var n3;
    n3 = n1+n2;
    return n3; //retorno el valor
}
```

Funciones

```
func sumar(int n1, int n2){
    var n3;
    n3 = n1+n2;
    return n3; // retorno el valor
}


func holamundo(){
    print(“Hola mundo”);
}


func conversion(double pies, string tipo){
    if (tipo == “metro”) {
        return pies/3.281;
    }
    else {
        return -1;
    }
}
```

Función Print

```
print(“Hola mundo!!”)
print(“Sale compi \n” + valor);
print(suma(2,2))
```

Función Main

```
main(){
    print(“hola soy un mensaje”);
    // mas instruciones
}
```