Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Organización de Lenguajes y Compiladores 2

Guillermo Alfredo Peitzner Estrada – 201504468.

Manual de Usuario

Augus

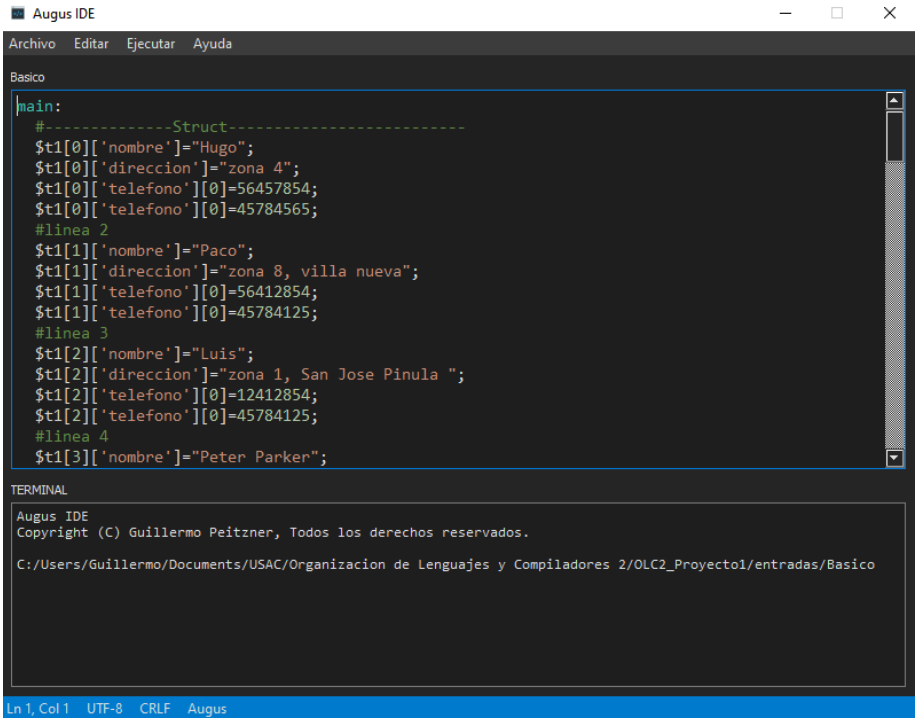
Augus es un lenguaje de programación, basado en PHP y en MIPS. Su principal funcionalidad es ser un lenguaje intermedio, ni de alto nivel como PHP ni de bajo nivel como el lenguaje ensamblador de MIPS.

El lenguaje tiene dos restricciones: la primera, es que cada instrucción es una operación simple; y la segunda, es que en cada instrucción hay un máximo de dos operandos y su asignación (si la hubiera).

Es un lenguaje débilmente tipado, sin embargo, si se reconocen cuatro tipos de datos no explícitos: entero, punto flotante, cadena de caracteres y arreglo.

Para manejar el flujo de control se proporciona la declaración de etiquetas, sin tener palabras reservadas para ese uso. Es decir, no hay ciclos for, while, ni do-while.

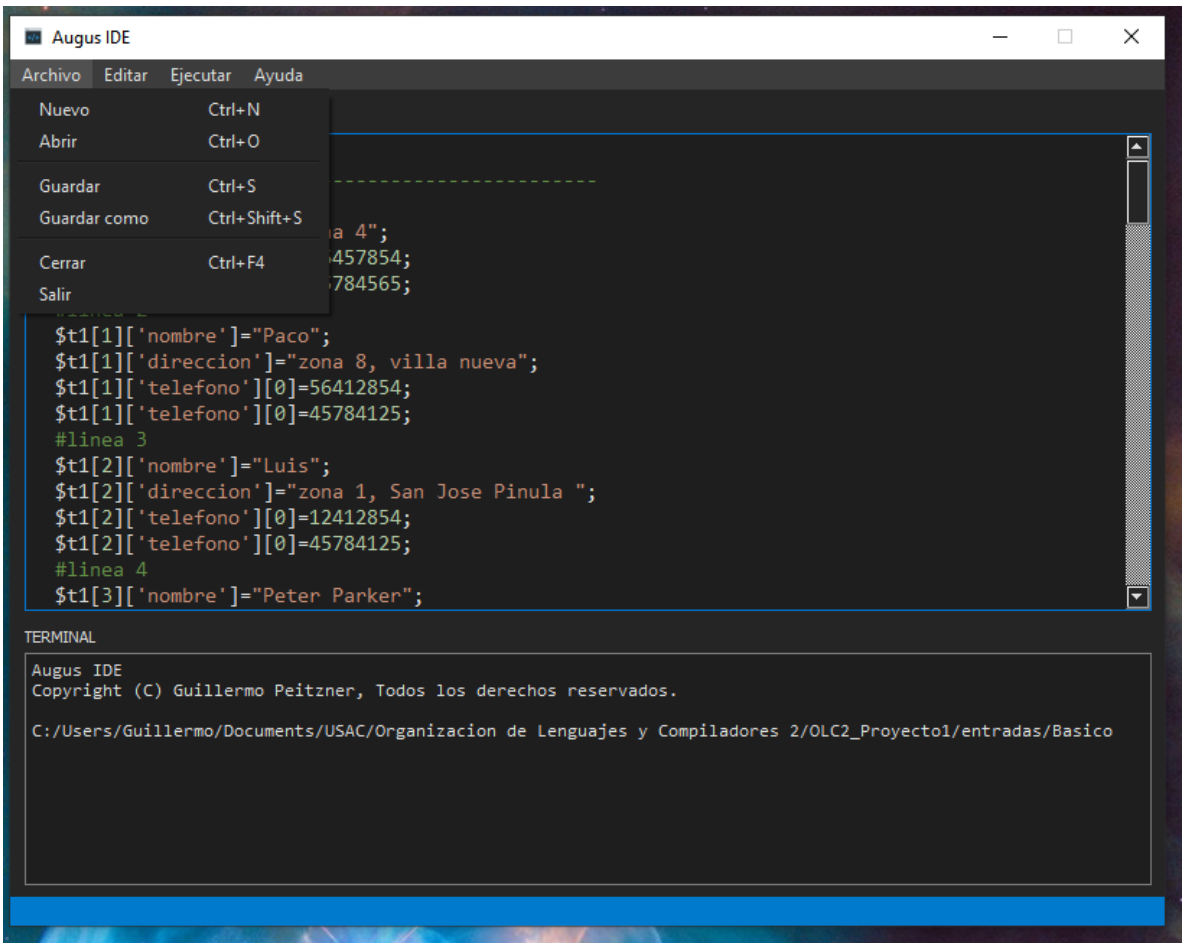
IDE



Archivo

En este menú se cuenta las siguientes opciones:

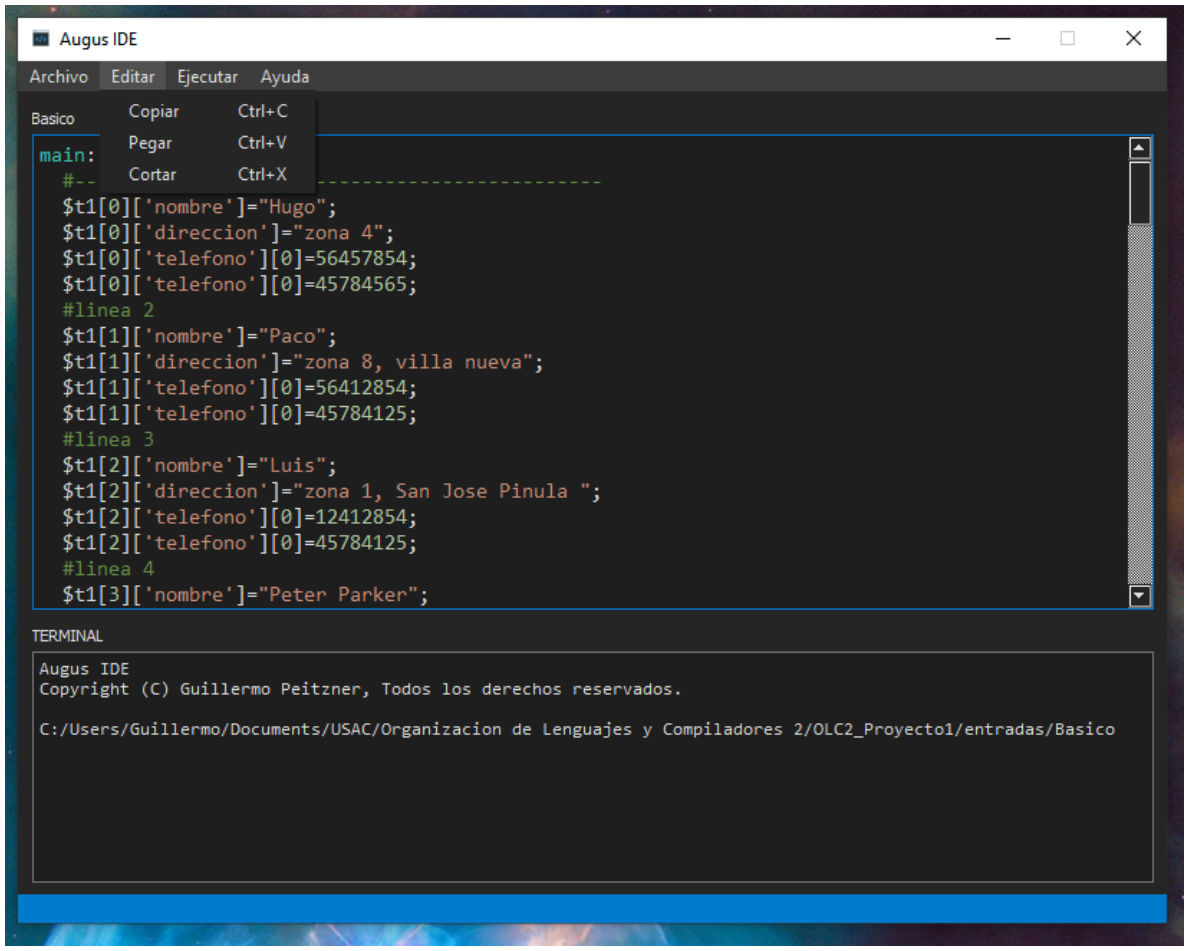
1. Nuevo: crea un nuevo archivo.
2. Abrir: abre un archivo.
3. Guardar: guarda el documento actual.
4. Guardar como: guarda el documento actual estableciendo la ruta destino.
5. Cerrar: cierra el documento actual.
6. Salir: cierra el IDE.



Editar

Este menú cuenta con las siguientes opciones:

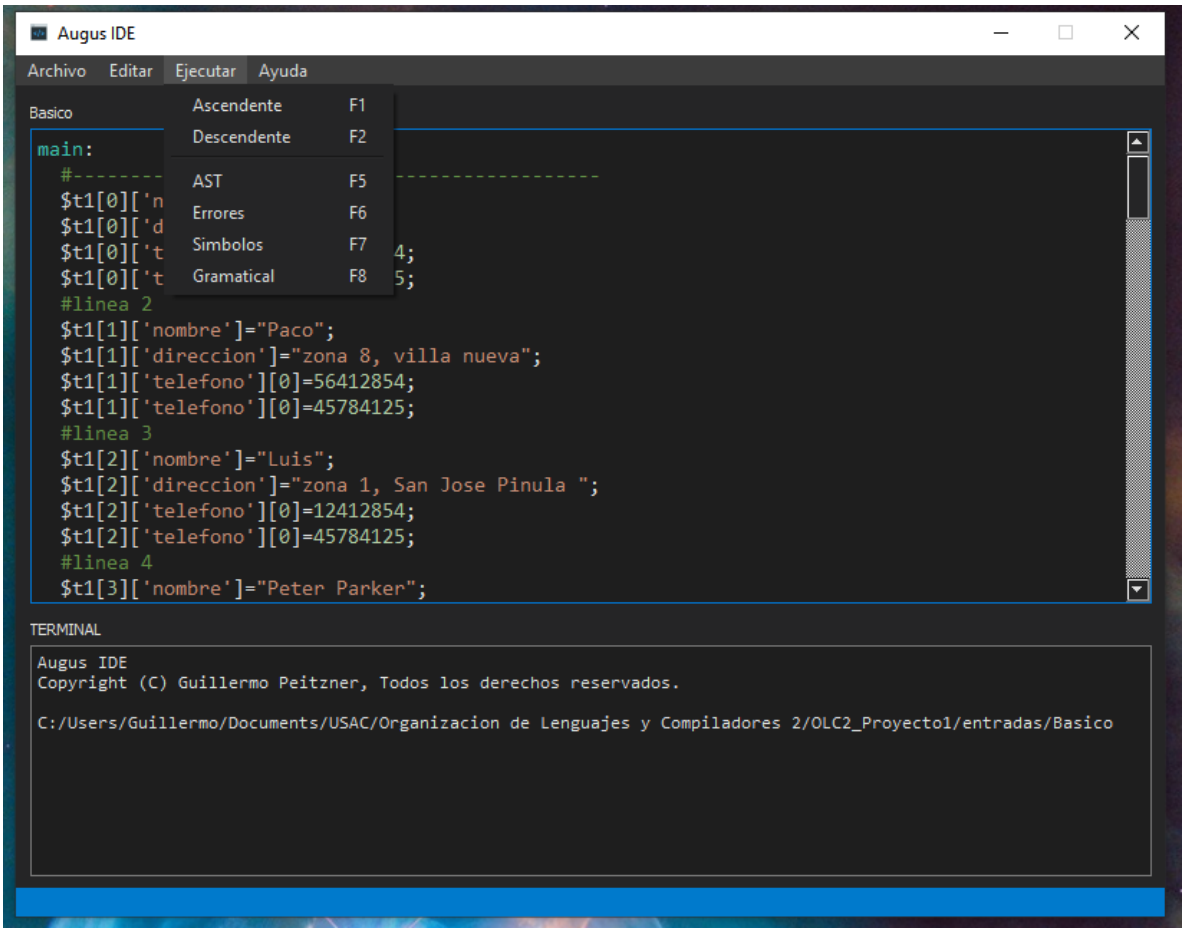
1. Copiar: copia el contenido del documento actual al portapapeles.
2. Pegar: pega el contenido del portapapeles en el documento actual.
3. Cortar: corta el contenido del documento actual al portapapeles.



Ejecutar

Este menú cuenta con las siguientes opciones:

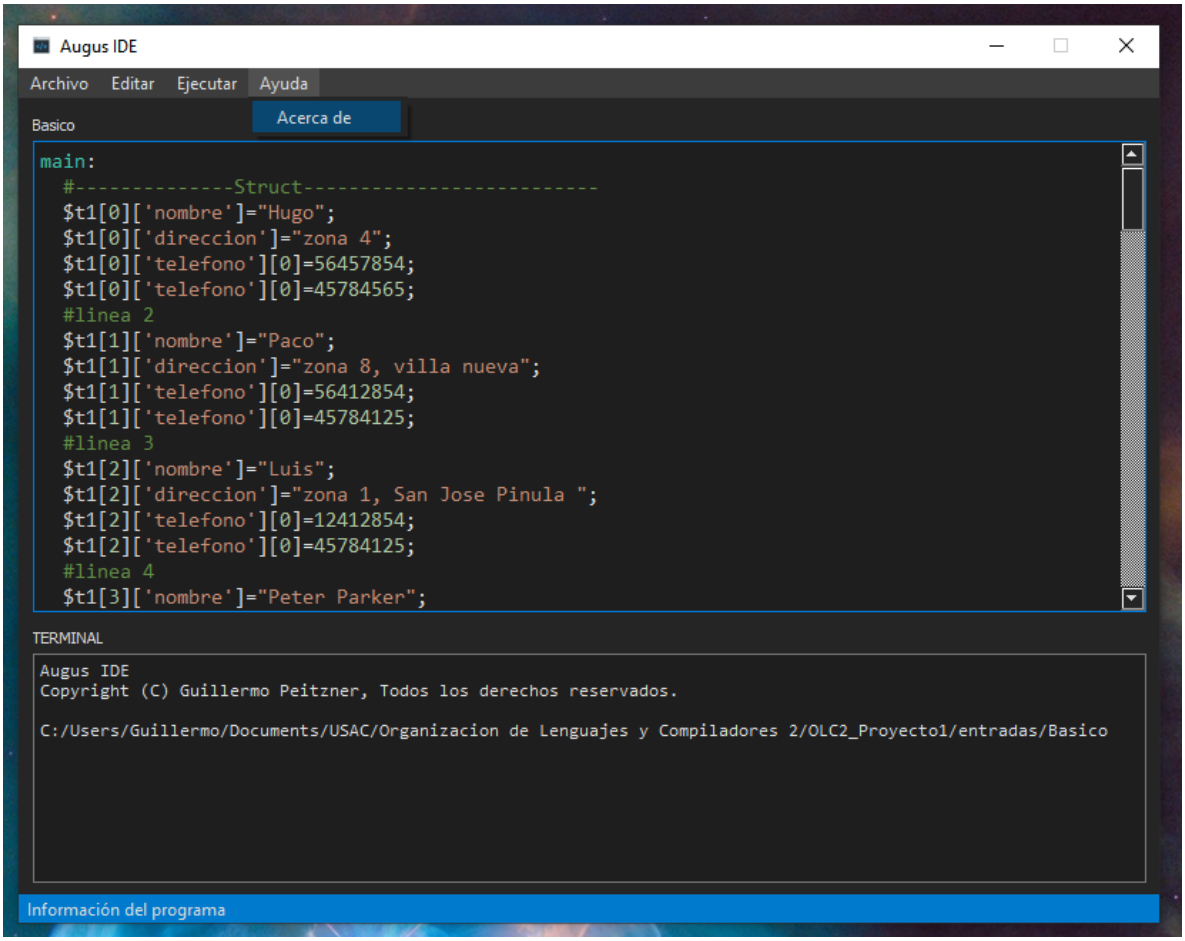
1. Ascendente: ejecuta el documento de forma ascendente.
2. Descendente: ejecuta el documento de forma descendente.
3. AST: muestra el contenido del árbol sintáctico en una imagen.
4. Errores: muestra errores léxicos y sintácticos en una página web.
5. Símbolos: muestra el contenido de la tabla de símbolos en una página web.
6. Gramatical: muestra la estructura gramatical del documento en una página web.



Ayuda

Este menú cuenta con las siguientes opciones:

1. Acerca de: muestra el repositorio fuente del programa.



Accesos por teclado

Todas las opciones mostradas anteriormente pueden ser accionadas mediante la combinación de teclas que se muestra en el respectivo menú de cada operación. Esto solo aplica para las opciones que tienen atajos.

Barra de Estado

En la barra de estado muestra información relevante de la posición actual del cursor dentro del programa.

Sintaxis de Augus

La sintaxis de Augus es similar al lenguaje PHP, pero con la salvedad de solo utilizar instrucciones simples, además de tener solamente las operaciones básicas que determina un lenguaje ensamblador similar a MIPS. El manejo de apuntadores también se basa en el comportamiento de PHP y no de C.

Definición de registros (variables)

$t0..$tn	Temporales
$a0..$an	Parámetros
$v0..$vn	Valores devueltos por funciones
$ra	Simulador de dirección de retorno por nivel
$s0..$sn	Pilas
$sp	Puntero de la pila

Instrucciones simples y unarias

main:	Inicio del programa. (obligatorio)
label:	Definición del inicio de una etiqueta.
goto label;	Salto incondicional hacia una etiqueta.
$t1 = 10;	Asignación numérica.
$t1 = 'hola'; $t1 = "hola";	Asignación de una cadena de caracteres.
$t1 = $t2;	Copia simple.
$t1 = - $t2;	Negativo.
$t1 = &$t2;	$t1 es un puntero a la dirección de $t2.
unset($t1);	Destruye la variable $t1.
print($t1);	Imprime en pantalla el contenido de $t1. Solo se pueden imprimir registros no operaciones.
$t1 = read();	Lee la entrada del teclado queda en $t1.
#comment	Comentario de una sola línea.
exit;	Finaliza la ejecución. (opcional)

Conversiones

$t1 = (int) $t2;	Si $t2 tiene decimales son eliminados. Si $t2 es un carácter se toma su código ASCII. Si $t2 es una cadena se toma el ASCII del primer carácter. Si $t2 es un arreglo se aplica al primer elemento las reglas anteriores.
$t1 = (float) $t2;	Si $t2 es entero se agrega ".0". Si $t2 es un carácter se toma su código ASCII con decimal. Si $t2 es una cadena se toma el ASCII del primer carácter más el decimal. Si $t2 es un arreglo se aplica al primer elemento las reglas anteriores.
$t1 = (char) $t2;	Si $t2 es un número de 0 a 255 se convierte en el carácter basado en ASCII. Si $t2 es un número mayor a 255 entonces se aplica el módulo 256 para extraer el ASCII. Si $t2 es un decimal, se quitan los decimales y se aplican las reglas anteriores. Si $t2 es una cadena, se almacena solo el primer carácter. Si $t2 es un arreglo, se almacena solo el primer valor aplicando las reglas anteriores.

Instrucciones aritméticas

$t1 = (int) $t2;	Si $t2 tiene decimales son eliminados. Si $t2 es un carácter se toma su código ASCII. Si $t2 es una cadena se toma el ASCII del primer carácter. Si $t2 es un arreglo se aplica al primer elemento las reglas anteriores.
$t1 = (float) $t2;	Si $t2 es entero se agrega ".0". Si $t2 es un carácter se toma su código ASCII con decimal. Si $t2 es una cadena se toma el ASCII del primer carácter más el decimal. Si $t2 es un arreglo se aplica al primer elemento las reglas anteriores.
$t1 = (char) $t2;	Si $t2 es un número de 0 a 255 se convierte en el carácter basado en ASCII. Si $t2 es un número mayor a 255 entonces se aplica el módulo 256 para extraer el ASCII. Si $t2 es un decimal, se quitan los decimales y se aplican las reglas anteriores. Si $t2 es una cadena, se almacena solo el primer carácter. Si $t2 es un arreglo, se almacena solo el primer valor aplicando las reglas anteriores.

Instrucciones lógicas

$t1 = !$t2;	Not, si $t2 es 0 $t1 es 1, si $t2 es 1 $t1 es 0.
$t1 = $t2 && $t3;	And, 1 para verdadero, 0 para falso.
$t1 = $t2 || $t3;	Or, 1 para verdadero, 0 para falso.
$t1 = $t2 xor $t3;	Xor, 1 para verdadero, 0 para falso.

Instrucciones bit a bit:

$t1 = ~$t2;	Not.
$t1 = $t2 & $t3;	And.
$t1 = $t2 | $t3;	Or.
$t1 = $t2 ^ $t3;	Xor.
$t1 = $t2 << $t3;	Shift de $t2, $t3 pasos a la izquierda.
$t1 = $t2 >> $t3;	Shift de $t2, $t3 pasos a la derecha.

Instrucciones relacionales

$t1 = $t2 == $t3;	$t1 = 1 si $t2 es igual a $t3, sino 0.
$t1 = $t2 != $t3;	$t1 = 1 si $t2 no es igual a $t3, sino 0.
$t1 = $t2 >= $t3;	$t1 = 1 si $t2 es mayor o igual a $t3, sino 0.
$t1 = $t2 <= $t3;	$t1 = 1 si $t2 es menor o igual a $t3, sino 0.
$t1 = $t2 > $t3;	$t1 = 1 si $t2 es mayor a $t3, sino 0.
$t1 = $t2 < $t3;	$t1 = 1 si $t2 es menor a $t3, sino 0.

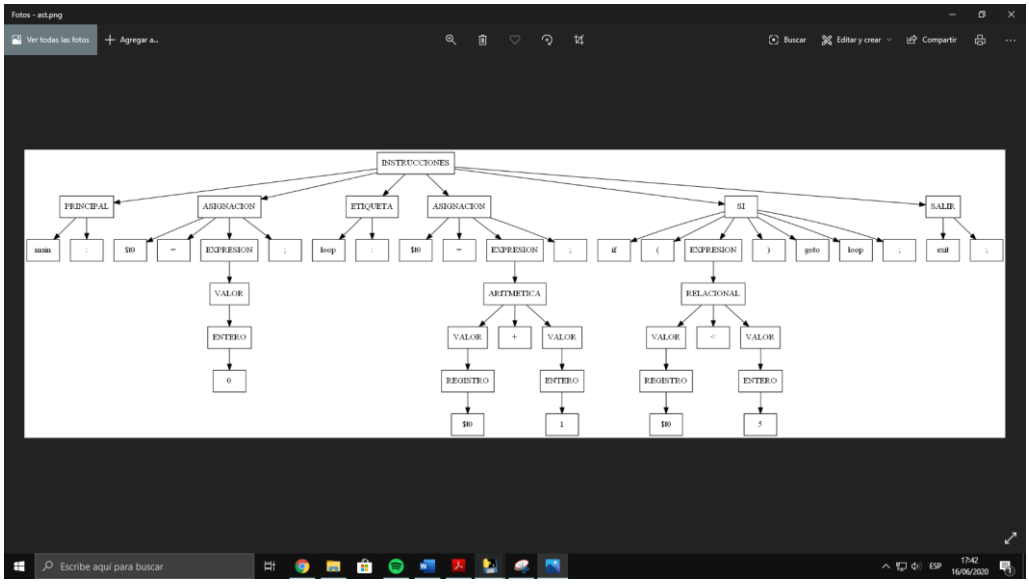
Arreglos, cadenas y structs

$t1 = array();	Define $t1 como un arreglo o un struct, para diferenciarlos se utiliza ya sea el valor numérico o el nombre asociativo.
$t1[4] = 1;	Asignación de un valor numérico (1) a un índice del arreglo (4).
$t1['nombre'] = 'carlos'; $t1["nombre"] = "carlos";	Asignación de un valor cadena (carlos) a un componente del struct (nombre).
$t1 = $t1[4];	Acceso a un índice del arreglo.
$t1 = $t2['nombre'];	Acceso a un componente del struct.
$t1 = 'hola'; print($t1[0]); #imprime h	Acceder a un carácter de una cadena.

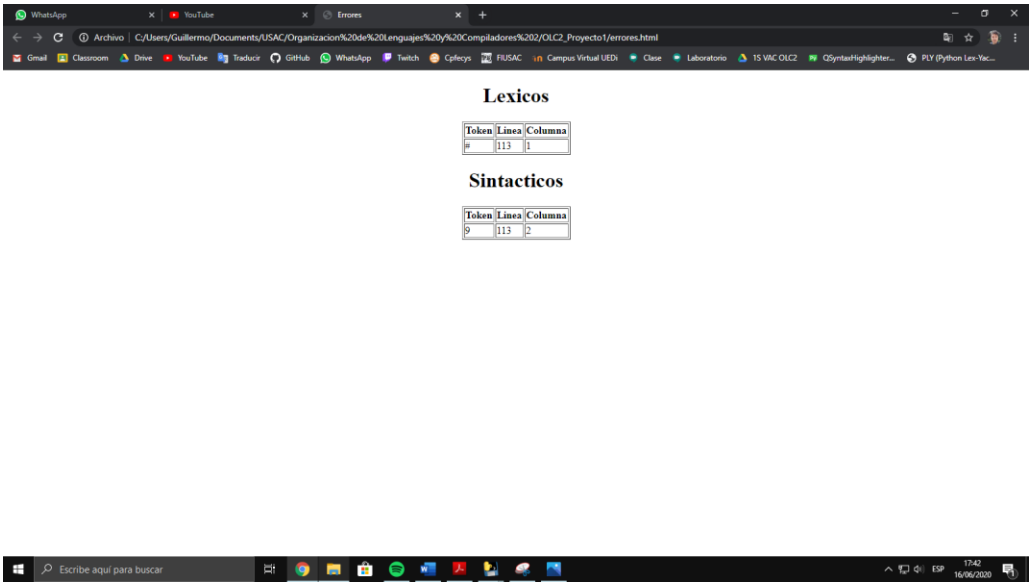
Instrucciones de control

if ($t1) goto label;	Salto condicional, si $t1 es 1 salto, sino sigue la siguiente instrucción, $t1 puede ser cualquier tipo de instrucción simple. Las estructuras de control se implementan utilizando este salto condicional.

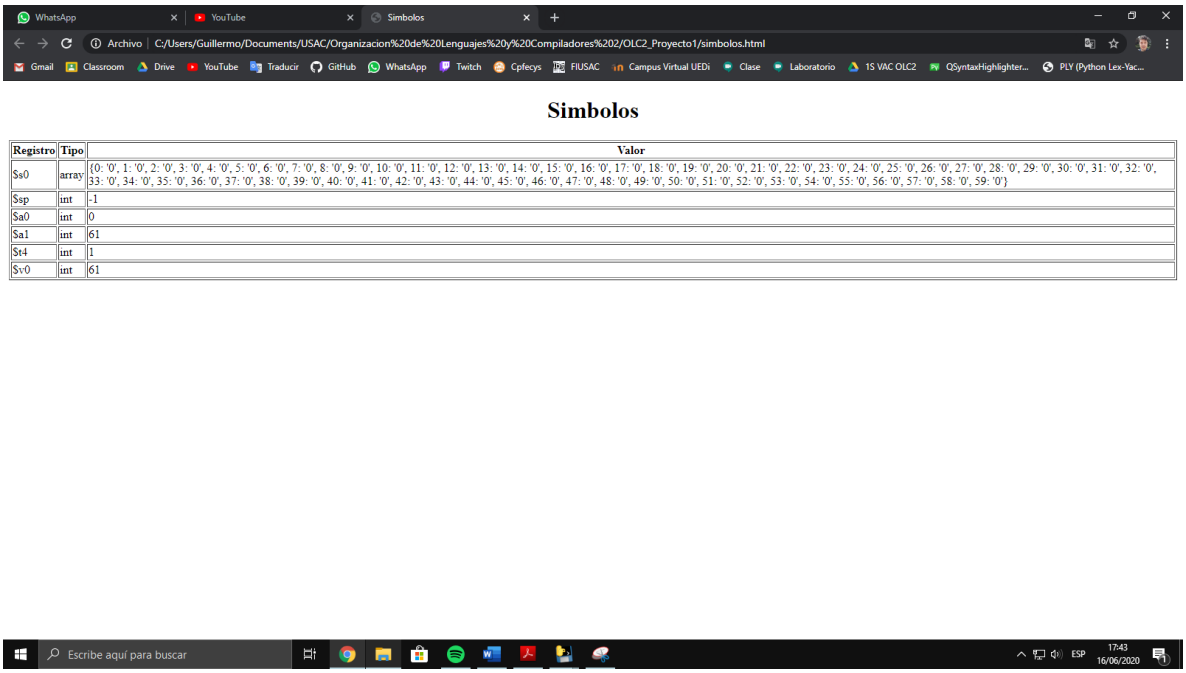
AST



Errores



Símbolos



Simbolos

Registro	Tipo	Valor
$s0	array	{0: '0', 1: '0', 2: '0', 3: '0', 4: '0', 5: '0', 6: '0', 7: '0', 8: '0', 9: '0', 10: '0', 11: '0', 12: '0', 13: '0', 14: '0', 15: '0', 16: '0', 17: '0', 18: '0', 19: '0', 20: '0', 21: '0', 22: '0', 23: '0', 24: '0', 25: '0', 26: '0', 27: '0', 28: '0', 29: '0', 30: '0', 31: '0', 32: '0', 33: '0', 34: '0', 35: '0', 36: '0', 37: '0', 38: '0', 39: '0', 40: '0', 41: '0', 42: '0', 43: '0', 44: '0', 45: '0', 46: '0', 47: '0', 48: '0', 49: '0', 50: '0', 51: '0', 52: '0', 53: '0', 54: '0', 55: '0', 56: '0', 57: '0', 58: '0', 59: '0'}
$sp	int	-1
$a0	int	0
$a1	int	61
$t4	int	1
$v0	int	61

Gramatical

Gramatical

Produccion	Regla
EXPRESION -> array()	EXPRESION.val = OperacionFuncion(array, VALOR.val)
ASIGNACION -> $s0 = EXPRESION;	ASIGNACION.val = Asignacion($s0, EXPRESION.val, t.lineno)
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
VALOR -> 1	VALOR.val = Entero(1)
EXPRESION -> - VALOR	EXPRESION.val = OperacionCasteo(-, VALOR.val)
ASIGNACION -> $sp = EXPRESION;	ASIGNACION.val = Asignacion($sp, EXPRESION.val, t.lineno)
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
VALOR -> 3	VALOR.val = Entero(3)
EXPRESION -> VALOR	EXPRESION.val = VALOR.val
ASIGNACION -> $a0 = EXPRESION;	ASIGNACION.val = Asignacion($a0, EXPRESION.val, t.lineno)
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
VALOR -> 3	VALOR.val = Entero(3)
EXPRESION -> VALOR	EXPRESION.val = VALOR.val
ASIGNACION -> $a1 = EXPRESION;	ASIGNACION.val = Asignacion($a1, EXPRESION.val, t.lineno)
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
VALOR -> $sp	VALOR.val = Registro($sp)
VALOR -> 1	VALOR.val = Entero(1)
EXPRESION -> VALOR + VALOR	EXPRESION.val = OperacionAritmetica(VALOR.val,+, VALOR.val)
ASIGNACION -> $sp = EXPRESION;	ASIGNACION.val = Asignacion($sp, EXPRESION.val, t.lineno)
INSTRUCCION -> ASIGNACION	INSTRUCCION.val = ASIGNACION.val
VALOR -> $sp	VALOR.val = Registro($sp)
EXPRESION -> VALOR	EXPRESION.val = VALOR.val
ACCESO -> [EXPRESION]	ACCESO.val = EXPRESION.val
ACCESOS_P -> None	ACCESOS_P.val = new list()
ACCESOS -> ACCESO INSTRUCCION_P	ACCESOS_P.add(ACCESO); ACCESOS_P.reverse(); ACCESOS.val = ACCESOS_P
VALOR -> $a0	VALOR.val = Registro($a0)
EXPRESION -> VALOR	[illegible]ON.val = VALOR.val