Manual de Usuario

PROYECTO 2 - COMPILADORES 2

Ricardo Menéndez Tobías
201602916 |

Contenido

MinorC

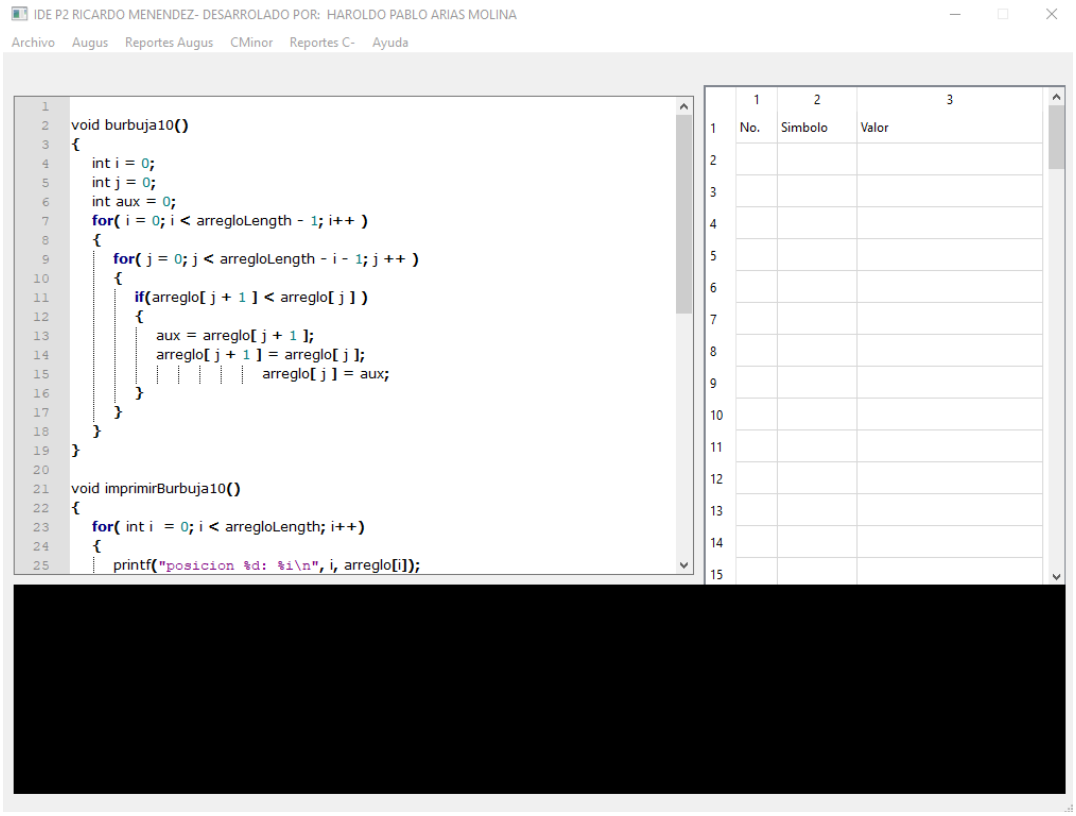
MinorC es un lenguaje de programación que es una versión simplificada de C. Este es capaz de realizar diferentes ciclos, sentencias de control, manejo de ámbitos, estructuras, arreglos y llamado a métodos. Su creación fue con la finalidad de poder dar un lenguaje fuente, el cual se pueda traducir a Augus, un lenguaje que simula el código intermedio.

Requisitos Mínimos

- Windows 7 en Adelante
- 2GB RAM
- Procesador Dual Core

IDE

Se tiene un IDE especial para la compilación y edición del código y la revisión de los reportes.



Módulos

Archivo

- Nuevo Archivo: Crea un archivo.
- Abrir: Abre un archivo
- Guardar: Guarda el archivo.
- Guardar Como: Guarda el archivo en una ruta seleccionada siempre.
- Salir: Cierra el programa.

Funciones de Archivo

- Cortar: Corta una porción de texto y la pone en el portapapeles.
- Copiar: Copia una porción de texto al portapapeles.
- Pegar: Pega el contenido del portapapeles.
- Buscar: Busca un texto y lo resalta.
- Seleccionar Todo: Selecciona todo el texto.

Augus

- Compilar Ascendente: Compila el texto con el analizador ascendente.
- Compilar Descendente: Compila el texto con el analizador descendente.
- Compilar Debug: Compila el texto con el analizador ascendente pero debugeando línea por linea.

Reportes Augus

- Gramatical Ascendente: Muestra un reporte con las producciones utilizadas en ejecución de la compilación ascendente.
- Gramatical Descendente: Muestra un reporte con las producciones utilizadas en ejecución de la compilación descendente.
- AST: Muestra el PDF que genera el árbol abstracto de análisis sintáctico.
- Tabla de Símbolos: Muestra un HTML con los símbolos creados durante la ejecución.
- Errores: Muestra los errores léxico-sintacticos-semanticos.

Minor C

- Compilar: Compila el texto con el analizador ascendente de Minor C.
- Cargar Codigo Augus: Carga al editor el código augus generado.
- Cargar Codigo MinorC: Carga al editor el código C usado.

Reportes Minor C

- Gramatical Ascendente: Muestra un reporte con las producciones utilizadas en ejecución de la compilación ascendente.
- AST: Muestra el PDF que genera el árbol abstracto de análisis sintáctico.
- Tabla de Símbolos: Muestra un HTML con los símbolos creados durante la ejecución.
- Errores: Muestra los errores léxico-sintacticos-semanticos.

Reportes:

Gramatical:

Contenido Etiqueta main	

ListaInstrucciones => ListaInstrucciones PC ";"
| Instrucciones PC ";"
	if(len(t)==3):
	t[0] = []
	t[0].append(t[1])
	
	if(len(t)==4):
	t[0] = t[1]
	t[0].append(t[2])

Intrucciones => asignacion | iff | jump | printt | ext | uns	t[0] = t[1]
Contenido ASIGNACION	
asignacion => var2 IGUAL exp	t[0] = Asignacion(t[1],t[3])

var2 => VAR arr
arr
	if(len(t)==3):
	t[0] = NodoVariable(t[1],t[2])
	else:
	t[0] = NodoVariable(t[1],None)

arr => arr IZQLLAVE E DERLLAVE
| IZQLLAVE E DERLLAVE
	if(len(t)==5):
	lar = t[1]
	lar.append(t[3])
	t[0] = lar
	else:
	lar = []
	lar.append(t[2])
	t[0] = lar

Contenido ARRAY	
arry => ARRAY LEFTPAR "(" DERPAR ")"	Se crea un objeto de tipo Array.

En este se muestran las producciones que se utilizaron en la ejecución del codigo.

AST

En este reporte, se muestra el árbol abstracto de análisis sintactico. Representando el código ingresado.

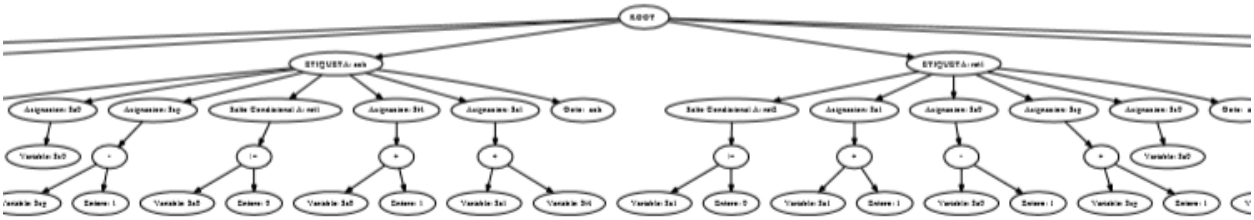


Tabla de Símbolos

Muestra las variables declaradas con sus valores junto a los métodos declarados en la ejecución.

Variables:

Variable	Valor	Tipo
$s0	{0: , 1: , 2: , 3: , 4: , 5: , 6: , 7: , 8: , 9: , 10: , 11: , 12: , 13: , 14: , 15: , 16: , 17: , 18: , 19: , 20: }	ARRAY
$sp	8	ENTERO
$a0	1	ENTERO
$a1	13	ENTERO
$t4	1	ENTERO

Metodos

Metodo
main
ret0
ack
ret1
ret2
ret3

Optimización de Codigo

El código intermedio generado (Augus) se generara con ciertas reglas aplicando opitmización de código, siendo mostrado luego en un reporte.

Lenguaje

El lenguaje permite al usuario crear código usando etiquetas, que vienen acompañadas de un cuerpo de instrucciones. Estas instrucciones terminan con un punto y coma (;).

Las instrucciones disponibles son:

Asignación	x = 1 + 1;
Declaración	int x = 1;
Salto	goto etiqueta;
If	If (1 < 2) { Cuerpo de Instrucciones} [else if (exp) { Cuerpo }]* [else { Cuerpo }]?
Métodos Nativos	printf, readf
for	for (int x = 0; x<10 ; x++) { Cuerpo }
while	while(expression) { Cuerpo }
do while	do { Cuerpo } while (expression)

Tipos de datos

- Enteros (int)
- Double
- FLoat
- Char

Además que Minor C permite el uso de arreglos y Structs.

Ejemplos de entradas:

```
int main(){
        int x = 0;
        int y = 1;
        printf(“Hola Mundo”);
}
```

****Para mas información de la sintaxis del programa, ver Archivo de: Gramaticas.**

```
 #-------------------Burbuja--------------

void burbuja10()
{
	int i = 0;
	int j = 0;
	int aux = 0;
	for( i = 0; i < arregloLength - 1; i++ )
	{
		for( j = 0; j < arregloLength - i - 1; j ++ )
		{
			if(arreglo[ j + 1 ] < arreglo[ j ] )
			{
				aux = arreglo[ j + 1 ];
				arreglo[ j + 1 ] = arreglo[ j ];
					arreglo[ j ] = aux;
			}
		}
	}
}

void imprimirBurbuja10()
{
	for( int i  = 0; i < arregloLength; i++)
	{
		printf("posicion %d: %i\n", i, arreglo[i]);
	}
}
```

```
int main()
{
int arreglo = { 658, 245, 654, 956, 5, 754, 100, 89, 98, 120};

int arregloLength = 10;
        printf("Arreglo Desordenado\n");
        imprimirBurbuja10();
        printf("--------------------------\n");
        burbuja10();
        printf("Arreglo Desordenado\n");
        imprimirBurbuja10();
}
```