

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 – MinorC IDE

Requisitos del sistema

Sistema operativo: Windows, Linux o Mac con Python 3.8 o superior y Graphviz en el path del sistema.

Procesador: Cualquier procesador con una frecuencia mayor a 1 GHz.

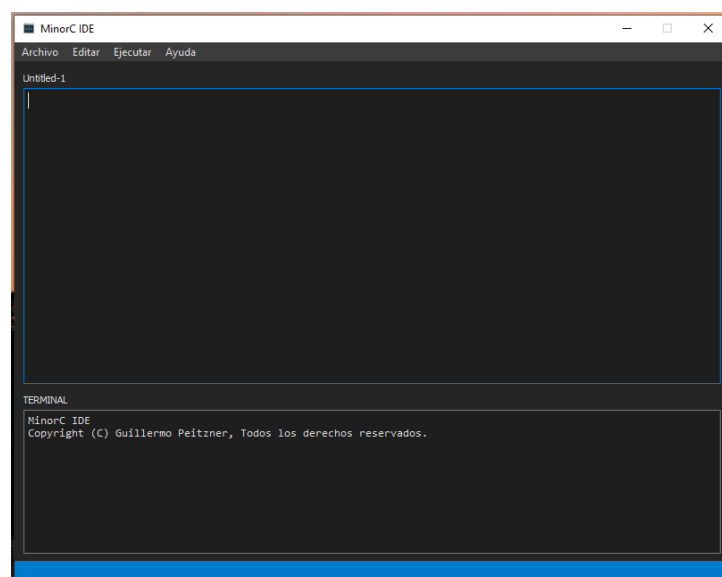
Memoria: 1 GB.

Gráficos: Integrados.

Espacio: 100 MB.

IDE

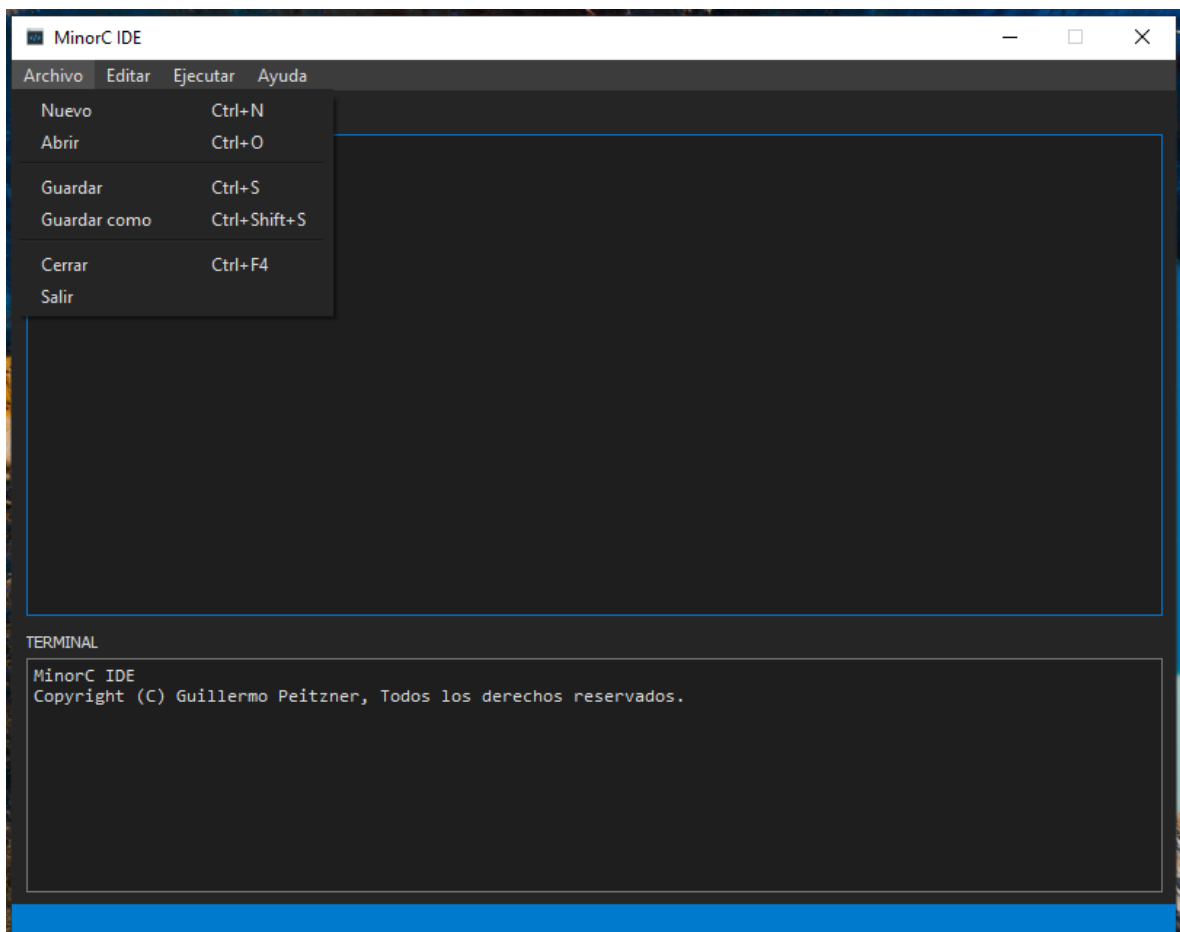
Entorno de desarrollo integrado MinorC, y su vista principal:



Archivo

Este menú contiene las funciones necesarias para la gestión del archivo del editor de código, cuenta con las siguientes opciones:

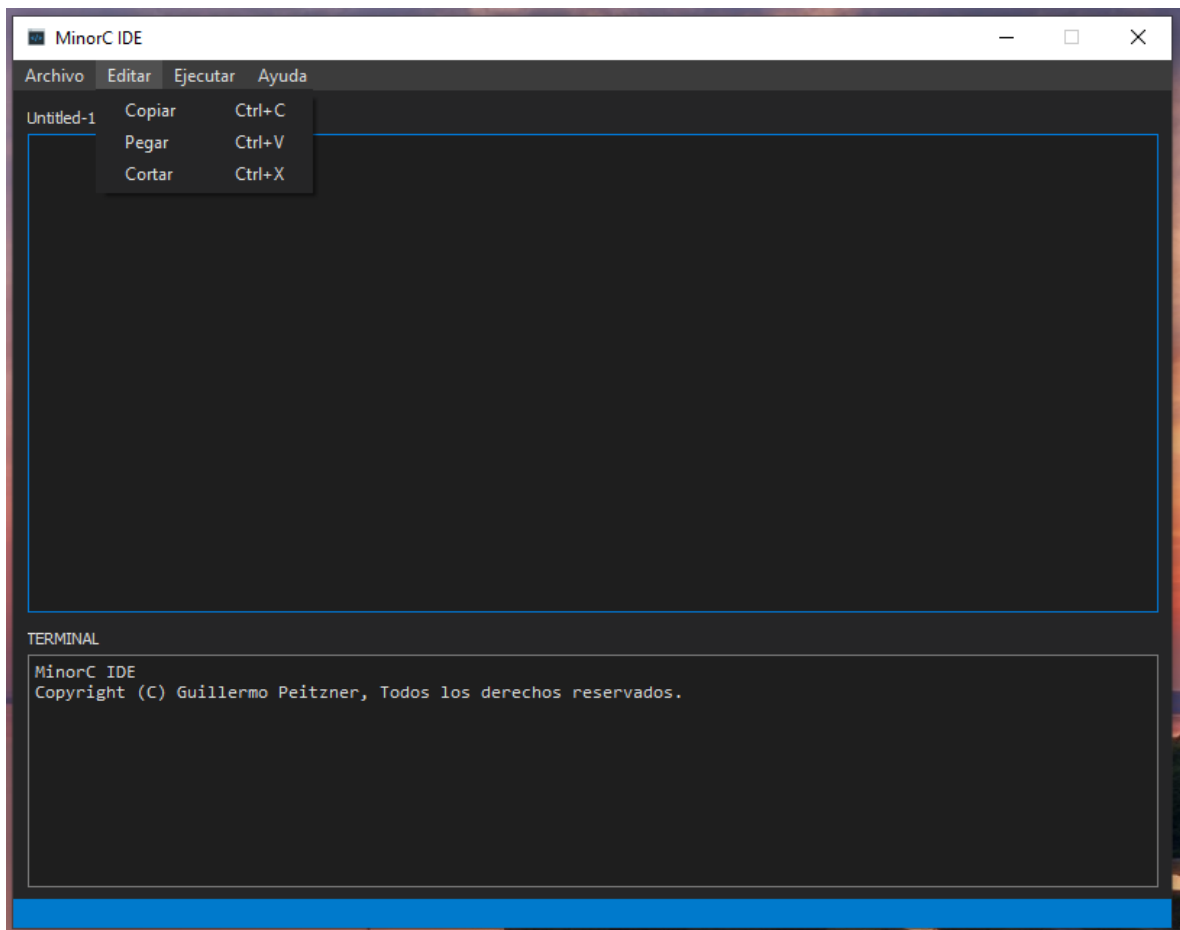
- Nuevo: Crea un nuevo archivo, limpiado el editor y la consola.
- Abrir: Abre un archivo y carga el contenido en el editor.
- Guardar: Guardar el archivo actual en su ruta actual y si no tiene una la solicita.
- Guardar como: Guardar el archivo actual especificando su ruta.
- Cerrar: Cierra el archivo actual del editor de código.
- Salir: Cierra el programa.



Editar

Este menú contiene las opciones necesarias para la gestión del código actual de trabajo, cuentan con las siguientes opciones:

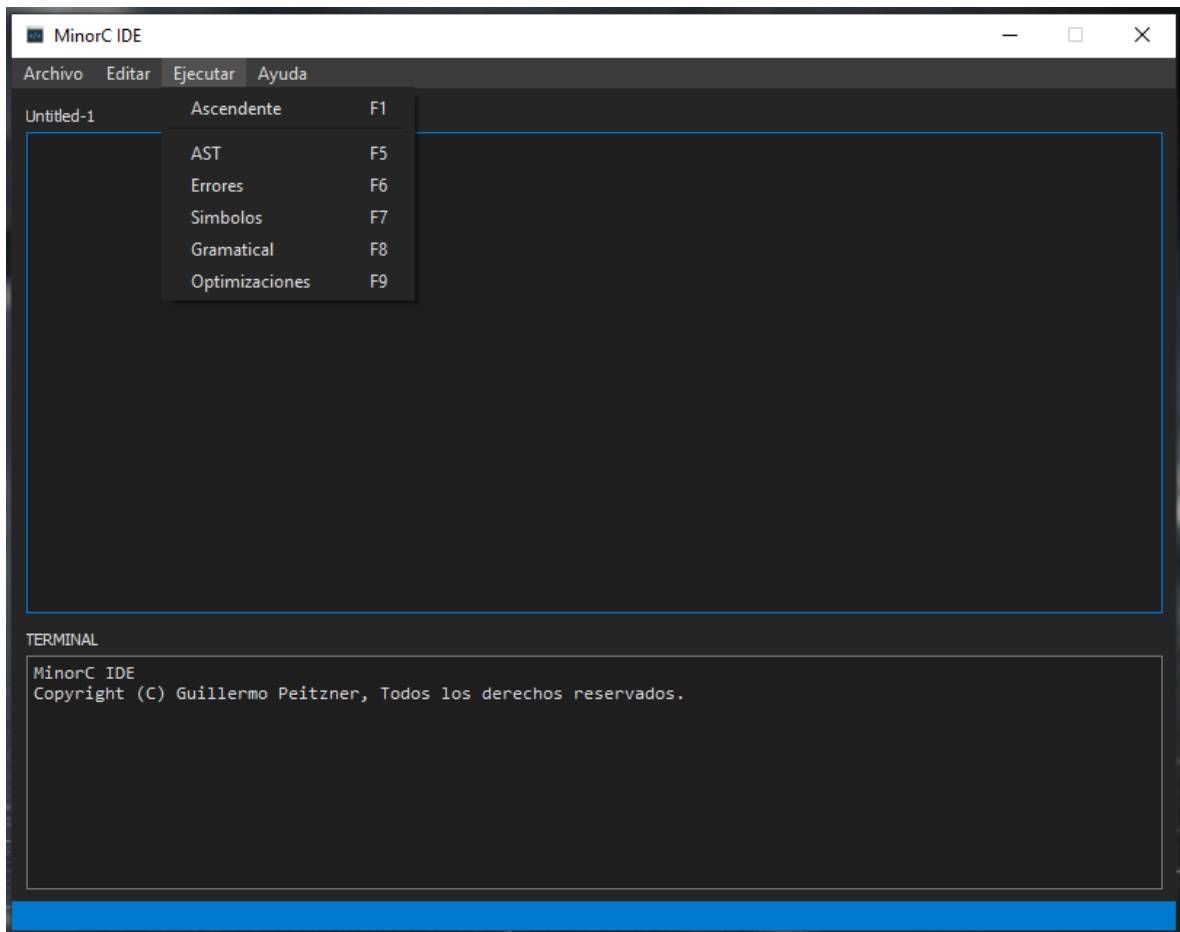
- Copiar: Esta opción permite copiar el contenido del editor actual al portapapeles.
- Pegar: Pega el contenido del portapapeles en el editor actual.
- Cortar: Corta el contenido del editor al portapapeles y limpia el editor de código.



Ejecutar

Este menú contiene todas las herramientas necesarias para la ejecución de código y generación de reportes, cuenta con las siguientes opciones:

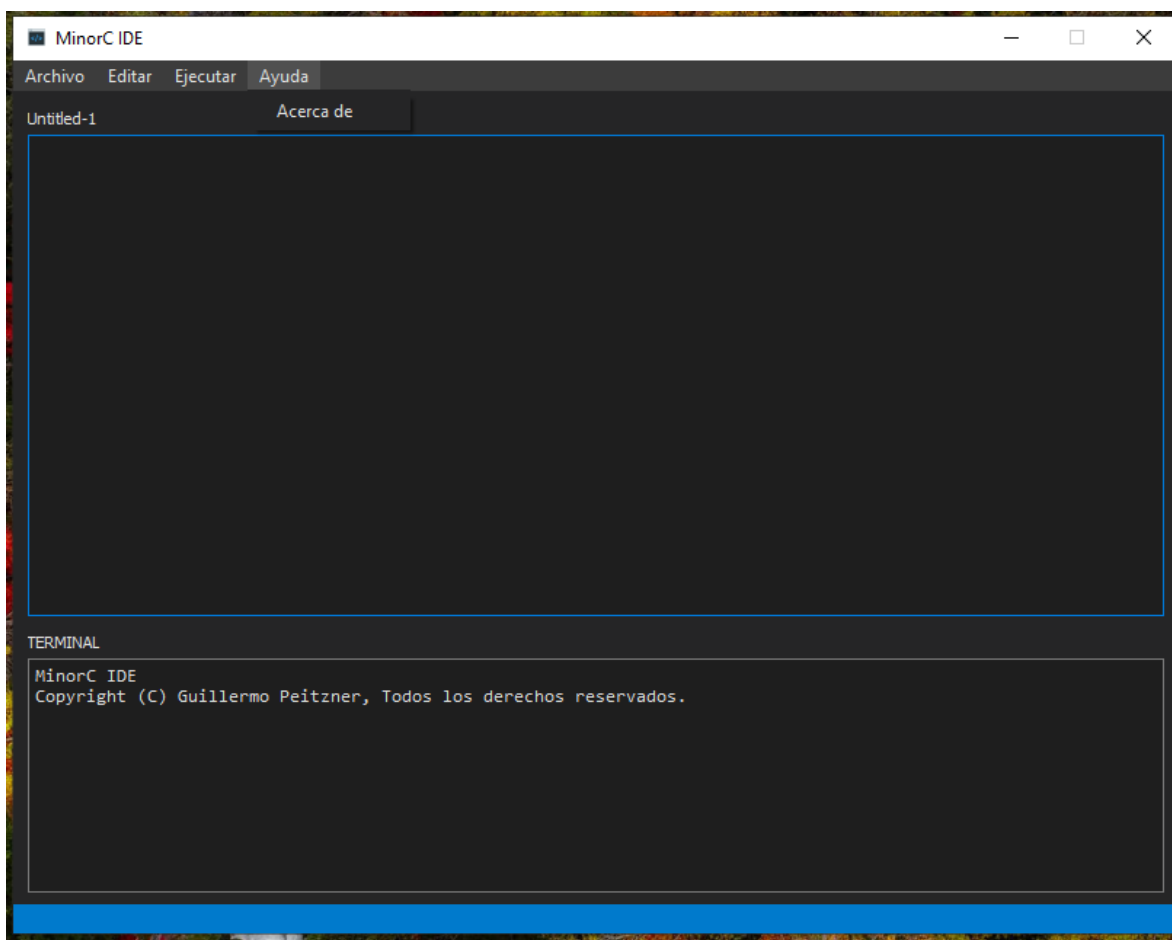
- Ascendente: hace un análisis ascendente del código del editor, construye un árbol sintáctico, genera código de tres direcciones y lo interpreta mostrando el resultado de la ejecución en la terminal. Luego de usar esta opción se pueden generar reportes.
- AST: Genera el árbol sintáctico de la entrada en el editor.
- Errores: Muestra un reporte en HTML con todos los errores léxicos y sintácticos del código actual.
- Símbolos: Muestra un reporte en HTML con todas las funciones y símbolos del código actual.
- Gramatical: Muestra un reporte en HTML con todas las reducciones en base a la gramática.
- Optimizaciones: Muestra tres reportes en HTML, el código de tres direcciones sin optimizar, el código optimizado y las reglas aplicadas para optimizar.



Ayuda

Este menú contiene información acerca del entorno de desarrollo integrado, cuenta con las siguientes opciones:

- Acerca de: muestra en la terminal un enlace al repositorio oficial del proyecto.



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 MinorC

MinorC es un subconjunto de C (cualquier ambigüedad de la sintaxis se arregla con la referencia de GNU C). MinorC está compuesto por los siguientes elementos:

Componentes léxicos

Identificadores: es secuencia de letras, y/o números, y/o guion bajo, que inicia con una letra. Distingue entre mayúsculas y minúsculas.

Palabras reservadas

Palabras reservadas: identificadores reservados. Entre algunos están: auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, register, return, sizeof, struct, switch, void y while.

Secuencias de escape

\\ Backlash, \' apostrophe, \" comilla doble, \n nueva línea, \r retroceso, y \t tabulación.

Tipos de datos

Tipos primitivos:

```
int i;  
char c = 'c';  
double d;  
float f;  
char s[] = "Hello";  
printf("%d %c %f %f %s", i, c, d, f, s);
```

Estructuras definidas por el programador, con las mismas reglas del struct:

```
struct punto  
{  
    int x,y;  
};  
  
struct punto mipunto;  
mipunto.x = 10;  
mipunto.y = 15;  
printf("%d %d \n", mipunto.x, mipunto.y);
```

Arreglos:

```
int m1[5];  
int m2[3] = {0, 1, 2};  
m1[0] = m2[2];  
int m3[2][2];  
struct punto puntos[3];
```

Expresiones y operadores

Una expresión es una instrucción con al menos un operador y cero o muchos operandos.

El operador de asignación (=) almacena valores en las variables, además de existir las asignaciones compuestas +=, -=, *=, /=, <=, %=, <<=, >>=, &=, ^=, |=.

Precedencia de operadores

Nivel	Operadores	Descripción	Asoci.
1	() [] -> .	Acceso a un elemento de un vector y paréntesis	Izquierdas
2	+ - ! ~ * & ++ -- (cast) sizeof	Signo (unario), negación lógica, negación bit a bit Acceso a un elemento (unarios): puntero y dirección Incremento y decremento (pre y post) Conversión de tipo (<i>casting</i>) y tamaño de un elemento	Derechas
3	* / %	Producto, división, módulo (resto)	Izquierdas
4	+ -	Suma y resta	Izquierdas
5	>> <<	Desplazamientos	Izquierdas
6	< <= >= >	Comparaciones de superioridad e inferioridad	Izquierdas
7	== !=	Comparaciones de igualdad	Izquierdas
8	&	Y (<i>And</i>) bit a bit (binario)	Izquierdas
9	^	O-exclusivo (<i>Exclusive-Or</i>) (binario)	Izquierdas
10		O (<i>Or</i>) bit a bit (binario)	Izquierdas
11	&&	Y (<i>And</i>) lógico	Izquierdas
12		O (<i>Or</i>) lógico	Izquierdas
13	?:	Condicional	Derechas
14	= *= /= %= += -= >>= <<= &= ^= =	Asignaciones	Derechas
15	,	Coma	Izquierdas

Operadores aritméticos

DdF	C	Descripción
+	+	Suma, adición
-	-	Resta, sustracción
.	*	Multiplicación, producto
Div	/	Cociente división entera
Mod	%	Resto división entera
/	/	División

Operadores relacionales

DdF	C	Descripción
>	>	Mayor
≥	>=	Mayor o igual
<	<	Menor
≤	<=	Menor o igual
=	==	Igual
≠	!=	Diferente

Operadores lógicos

DdF	C	Descripción
^	&&	And, y, conjunción
∨		Or, o, disyunción
¬	!	Not, no, negación

		Negación	Conjunción	Disyunción
		¬a	a ∧ b	a ∨ b
a	b	!a	a && b	a b
0	0	1	0	0
0	Cierto	1	0	1
Cierto	0	0	0	1
Cierto	Cierto	0	1	1

Operaciones bit a bit

		Negación	Conjunción	Disyunción
		$\neg a$	$a \wedge b$	$a \vee b$
a	b	$!a$	$a \&\& b$	$a b$
0	0	1	0	0
0	Cierto	1	0	1
Cierto	0	0	0	1
Cierto	Cierto	0	1	1

Operador Condicional

DdF	C	Descripción
$c ?$	$c ?$	Operador condicional $c ? e_1 : e_2$

Generalidades en expresiones

1. El operador `sizeof()` devuelve la cantidad en bytes del tipo de dato como argumento.
2. Los operadores de cambio de tipo (cast), pueden ser aplicados a los tipos de datos primitivos.
3. Las llamadas a funciones retornan un valor que puede ser utilizado como expresión.
4. El operador coma (,) sirve para separar expresiones, como en el caso de declaraciones, asignaciones e incluso en las partes del ciclo for.
5. Operador ternario, evalúa una expresión, si es verdadera retorna el segundo valor, de lo contrario devuelve el tercer valor. $a ? b : c$;
6. Precedencia de operadores, se evalúan de mayor a menor precedencia según la siguiente lista (algunos operadores pueden tener el mismo nivel) y se evalúan de izquierda a derecha: funciones, operador unario, casteo, `sizeof`, multiplicación, división, módulo, suma, resta, corrimiento de bits, relacionales, igual o diferente, AND, XOR, OR entre bits, AND y OR lógicos, operador ternario, asignaciones, y operador coma.

Declaraciones

1. Las etiquetas son identificadores de sección seguidos por el signo dos puntos, para ser utilizada la declaración goto.
2. Las declaraciones de expresiones son instrucciones que finalizan con el punto y coma.
3. La declaración condicional if, con la sintaxis if (condición) declaración else declaración. También se puede extender a multiples condiciones con else if.
4. La declaración switch, con la sintaxis: switch(condición){ case comparación declaración1 case comparación declaración2 default: declaración_default}.
5. La declaración while, es un ciclo con una condición de salida al inicio del ciclo. Con la sintaxis: while (condición) declaración.
6. La declaración do, es un ciclo con una condición de salida al final del ciclo. Con la sintaxis: do declaración while (condición);
7. La declaración for, es un ciclo con tres partes, una inicialización, otra comparación y una tercera de modificación de variable. Con la sintaxis: for(inicialización; condición; cambio) declaración.
8. Las anteriores declaraciones pueden escribirse de manera anidada, también existe una declaración nula, donde en una declaración se coloca solamente el punto y coma.
9. La declaración goto, es un salto no condicional que salta hacia una etiqueta.
10. La declaración break finaliza cualquier ciclo y el switch.
11. La declaración continue finaliza una iteración para continuar en la siguiente.
12. La declaración return finaliza la ejecución de una función, se puede agregar un valor de retorno.

Funciones

- La declaración de una función incluye su nombre, la lista de parámetros y un valor de retorno. La llamada a una función se puede hacer dentro de una declaración escribiendo el nombre de la función y los parámetros necesarios.
- También se define la función main como obligatoria, no teniendo ningún parámetro que definir, retornando un valor entero.
- Es permitido hacer funciones recursivas, y no se permiten hacer funciones anidadas.

Ejemplos

```
int main()
{
    int x = 0;
    int a = 0;
    while(x<4)
    {
        a = a + x;
        x = x + 1;
    }
    printf("%d",a);
}
```

```
int f2(int a)
{
    return a*a;
}

int f1(int a)
{
    return f2(a);
}
```

```
int main()
{
    int a = 5;
    a = f1(a);
    printf("%d",a);
}
```

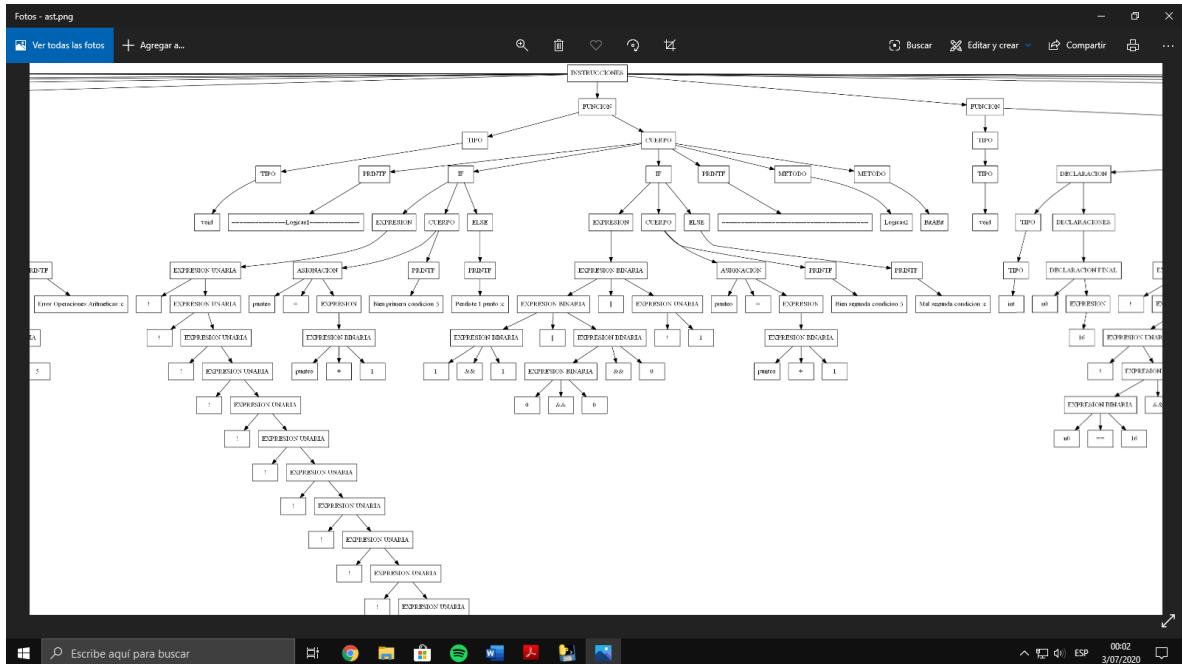
```
struct punto
{
    int x,y;
};

int main()
{
    punto lista[2];
    lista[0].x = 0;
    lista[0].y = 0;
    lista[1].x = 5;
    lista[1].y = 5;
    printf("%d %d\n",lista[0].x,lista[0].y);
    printf("%d %d\n",lista[1].x,lista[1].y);
}
```

Reportes generados

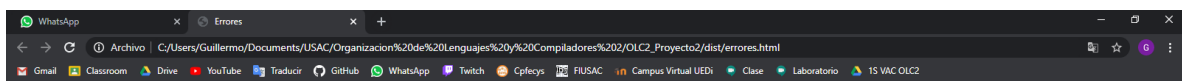
AST

Contiene la forma del árbol sintáctico generado:



Errores

Muestra los errores léxicos y sintácticos encontrados:



Lexicos

Token	Linea	Columna
int	7	55

Sintacticos

Token	Linea	Columna
int	8	5

Símbolos

Muestra las funciones y símbolos del archivo actual:

The screenshot shows a web browser window with the address bar displaying the path: `C:/Users/Guillermo/Documents/USAC/Organizacion%20de%20Lenguajes%20y%20Compiladores%202/OLC2_Proyecto2/dist/simbolos.html`. The page content includes two tables:

Funciones

Tipo	Identificador
void	Declaracion
void	Aritmeticas
void	operacionesBasicas
void	operacionesAvanzadas
void	Logicas
void	Logicas2
void	BitABit
void	Relacionales
void	relaciones1
void	relaciones2
int	main

Símbolos

Tipo	Identificador	Temporal
int	var1	\$t0
int	punteo	\$t1
int	var1	\$t2
char	str4	\$t5
double	db4	\$t6
double	db1	\$t7
char	chr4	\$t8

Gramatical

Muestra la estructura gramatical del código fuente:

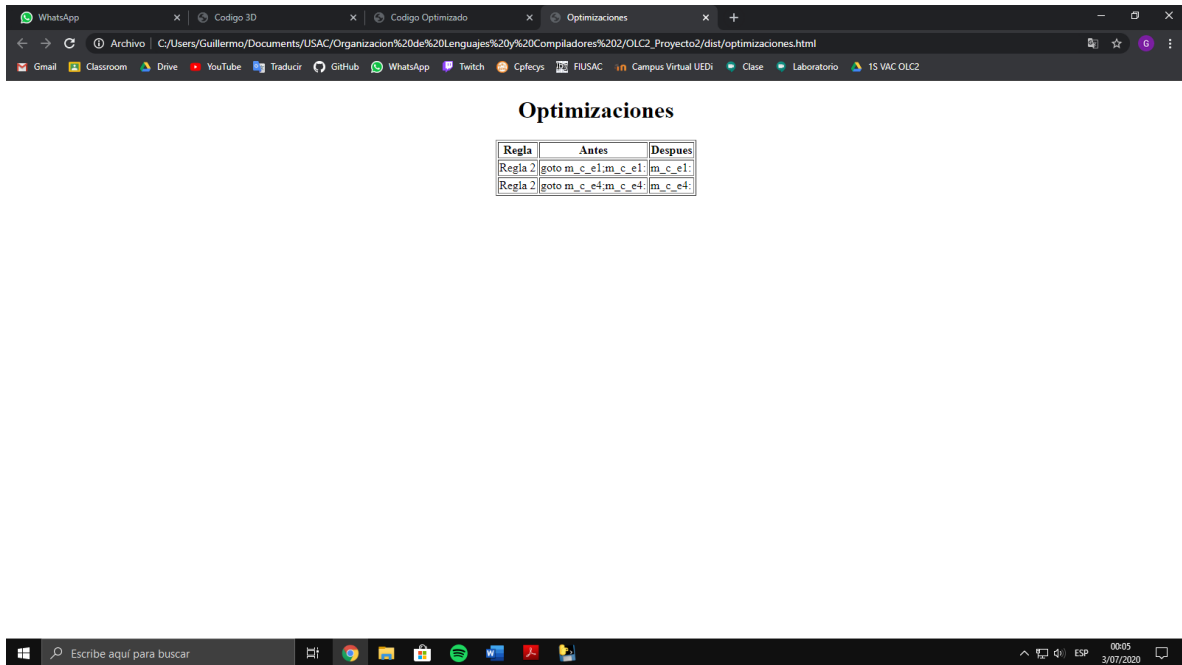
The screenshot shows a web browser window with the address bar displaying the path: `C:/Users/Guillermo/Documents/USAC/Organizacion%20de%20Lenguajes%20y%20Compiladores%202/OLC2_Proyecto2/dist/gramatical.html`. The page content includes a table with grammar rules:

Gramatical

Produccion	Regla
TIPO -> int char double float void	TIPO VAL = Tipo(int char double float void), None);
INDICES ->	INDICES VAL = NONE;
EXPRESION -> entero	EXPRESION VAL = entero;
DECLARACION_FINAL -> identificador INDICES = EXPRESION	DECLARACION_FINAL VAL = DeclaracionFinal(identificador, INDICES VAL, EXPRESIONES VAL, lineno);
LISTA_DECLARACION -> DECLARACION_FINAL	LISTA_DECLARACION VAL = Lista(DECLARACION_FINAL VAL);
DECLARACION -> TIPO LISTA_DECLARACION	DECLARACION VAL = Declaracion(TIPO VAL, LISTA_DECLARACION VAL);
INSTRUCCION_GLOBAL -> DECLARACION; ESTRUCTURA; FUNCION	INSTRUCCION_GLOBAL VAL = DECLARACION VAL ESTRUCTURA VAL FUNCION VAL;
LISTA_GLOBAL -> INSTRUCCION_GLOBAL	LISTA_GLOBAL VAL = Lista(INSTRUCCION_GLOBAL VAL);
TIPO -> int char double float void	TIPO VAL = Tipo(int char double float void), None);
INDICES ->	INDICES VAL = NONE;
EXPRESION -> entero	EXPRESION VAL = entero;
DECLARACION_FINAL -> identificador INDICES = EXPRESION	DECLARACION_FINAL VAL = DeclaracionFinal(identificador, INDICES VAL, EXPRESIONES VAL, lineno);
LISTA_DECLARACION -> DECLARACION_FINAL	LISTA_DECLARACION VAL = Lista(DECLARACION_FINAL VAL);
DECLARACION -> TIPO LISTA_DECLARACION	DECLARACION VAL = Declaracion(TIPO VAL, LISTA_DECLARACION VAL);
INSTRUCCION_GLOBAL -> DECLARACION; ESTRUCTURA; FUNCION	INSTRUCCION_GLOBAL VAL = DECLARACION VAL ESTRUCTURA VAL FUNCION VAL;
LISTA_GLOBAL -> LISTA_GLOBAL1 INSTRUCCION_GLOBAL	LISTA_GLOBAL1 ADD(INSTRUCCION_GLOBAL); LISTA_GLOBAL VAL = LISTA_GLOBAL1 VAL;
TIPO -> int char double float void	TIPO VAL = Tipo(int char double float void), None);
PARAMETROS ->	PARAMETROS VAL = NONE;
EXPRESION -> cadena	EXPRESION VAL = cadena;
LISTA_EXPRESIONES -> EXPRESION	LISTA_EXPRESIONES VAL = Lista(EXPRESION VAL);
PRINT -> printf (LISTA_EXPRESIONES)	PRINT VAL = Printf(LISTA_EXPRESIONES VAL, lineno);
INSTRUCCION_LOCAL -> ETIQUETA SALTO DECLARACION; ASIGNACION; IF SWITCH WHILE DO FOR PRINT; METODO;	INSTRUCCION_LOCAL VAL = ETIQUETA VAL SALTO VAL DECLARACION VAL ASIGNACION VAL IF VAL SWITCH VAL WHILE VAL DO VAL FOR VAL PRINT VAL METODO VAL;
LISTA_LOCAL -> INSTRUCCION_LOCAL	LISTA_LOCAL VAL = Lista(INSTRUCCION_LOCAL VAL);
TIPO -> int char double float void	TIPO VAL = Tipo(int char double float void), None);
INDICES ->	INDICES VAL = NONE;
EXPRESION -> entero	EXPRESION VAL = entero;

Optimizaciones

Muestra las optimizaciones realizadas en el código de tres direcciones:



The screenshot shows a web browser window with the title "Optimizaciones". The address bar displays the URL: `C:/Users/Guillermo/Documents/USAC/Organizacion%20de%20Lenguajes%20y%20Compiladores%202/OLC2_Proyecto2/dist/optimizaciones.html`. The browser's taskbar at the bottom shows various application icons, including WhatsApp, Classroom, Drive, YouTube, Traducir, GitHub, WhatsApp, Twitch, Cpfelgys, FIUSAC, Campus Virtual UED, Clase, Laboratorio, and 15 VAC OLC2. The main content area of the browser displays the following table:

Regla	Antes	Despues
Regla 2	<code>goto m_c_e1;m_c_e1</code>	<code>m_c_e1:</code>
Regla 2	<code>goto m_c_e4;m_c_e4</code>	<code>m_c_e4:</code>