Imagen que contiene dibujo, plato, taza

Descripción generada automáticamenteUniversidad 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.

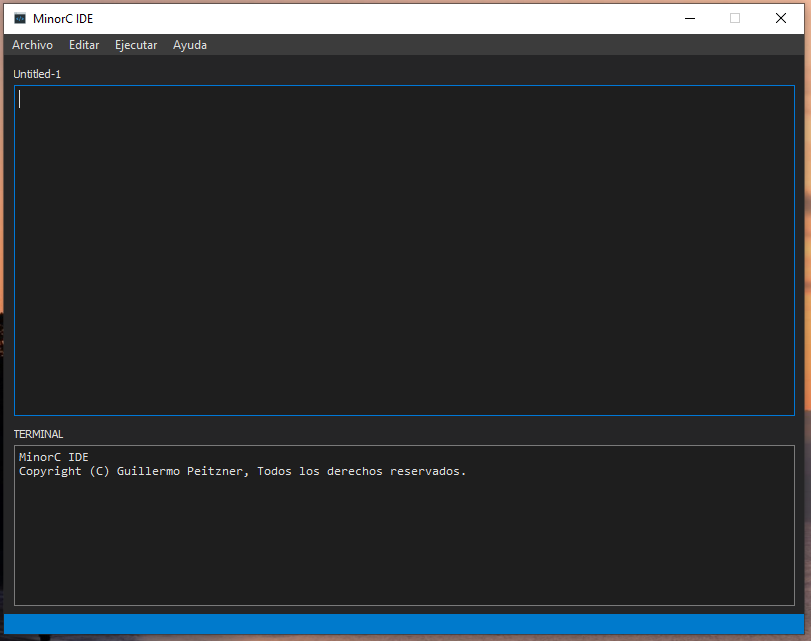
Memoria: 1 GB.

Gráficos: Integrados.

Espacio: 100 MB.

## IDE

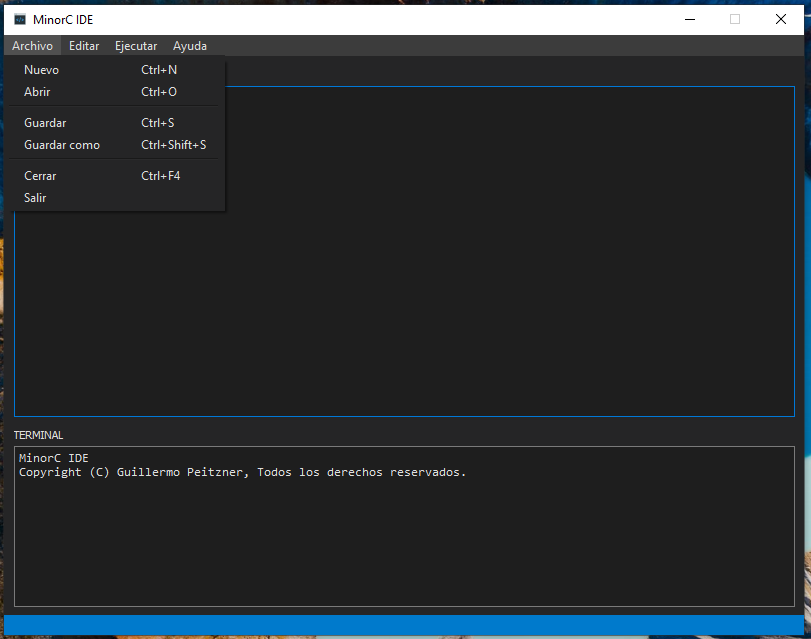
Entorno de desarrollo integrado MinorC, y su vista principal:



## Archivo

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

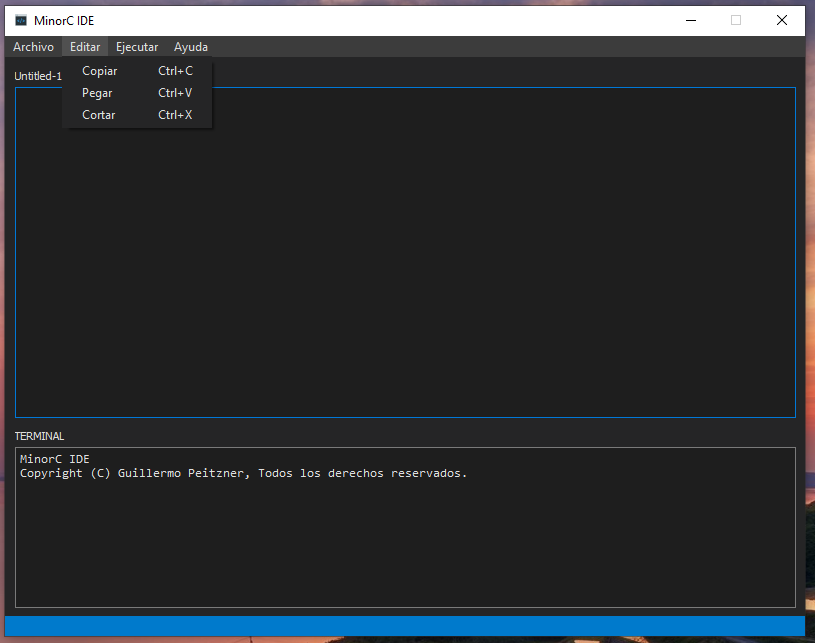
* Nuevo: Crea un nuevo archivo, limpiado el editor y la consola.
* Abrir: Abre un archivo y carga el contenido en el editor.
* Guardar: Guardar el archivo actual en su ruta actual y si no tiene una la solicita.
* Guardar como: Guardar el archivo actual especificando su ruta.
* Cerrar: Cierra el archivo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

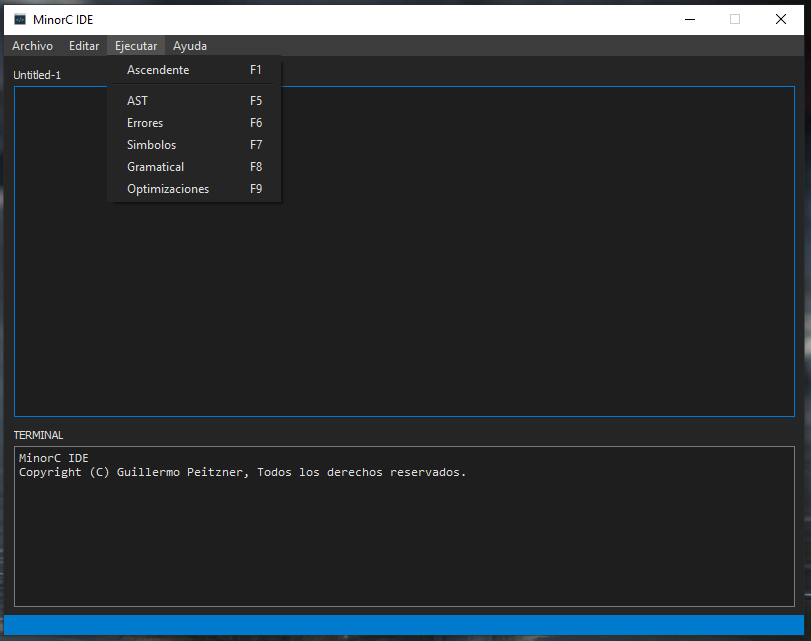
* Copiar: Esta opción permite copiar el contenido del editor actual al portapapeles.
* Pegar: Pega el contenido del portapapeles en el editor actual.
* Cortar: Corta el contenido del editor al portapapeles y limpia el editor de código.



## Ejecutar

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

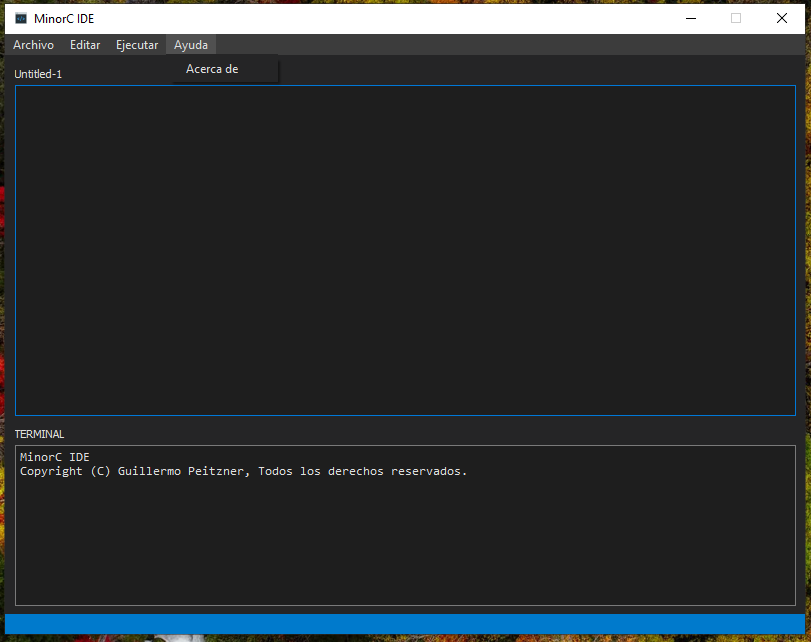
* Ascendente: hace un análisis ascendente del código del editor, construye un árbol sintáctico, genera código de tres direcciones y lo interpreta mostrando el resultado de la ejecución en la terminal. Luego de usar esta opción se pueden generar reportes.
* AST: Genera el árbol sintáctico de la entrada 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

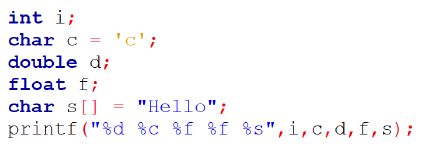
Palabras reservadas: identificadores reservados. Entre algunos están: auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, register, return, sizeof, struct, switch, void y while.

### Secuencias de escape

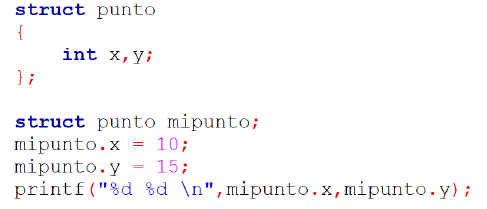
\\ Backlash, \’ apostrophe, \” comilla doble, \n nueva línea, \r retroceso, y \t tabulación.

### Tipos de datos

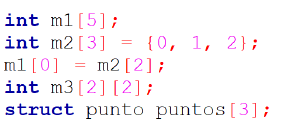
Tipos primitivos:



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



Arreglos:

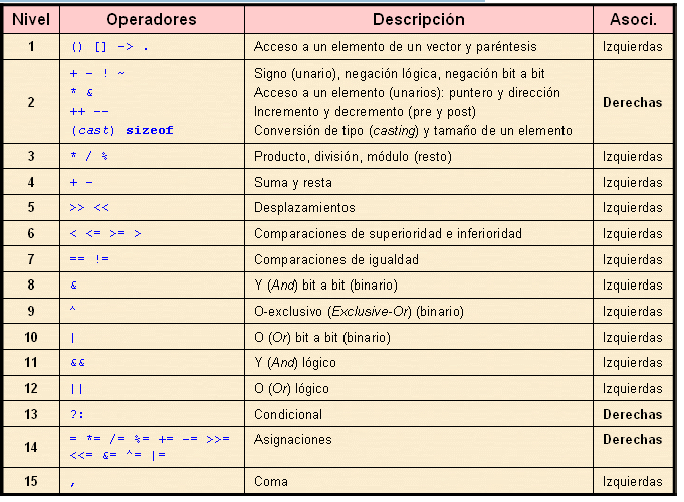


### Expresiones y operadores

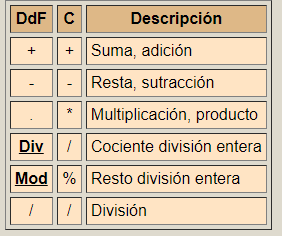
Una expresión es una instrucción con al menos un operador y cero o muchos operandos.

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

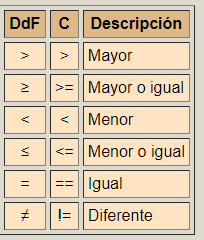
### Precedencia de operadores



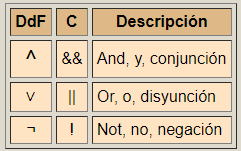
### Operadores aritméticos

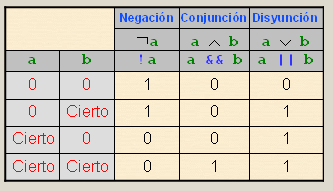


### Operadores relacionales



### Operadores lógicos

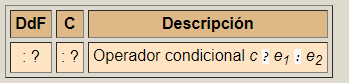




### Operaciones bit a bit



### Operador Condicional



### 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 evaluan de mayor a menor precedencia según la siguiente lista (algunos operadores pueden tener el mismo nivel) y se evaluan 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.

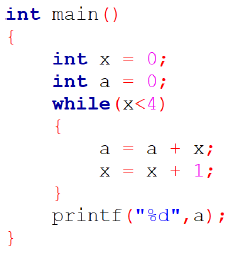
### Declaraciones

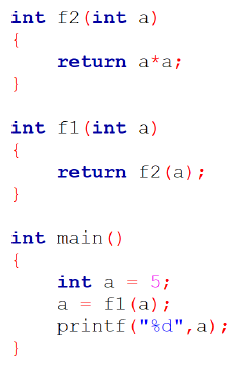
1. Las etiquetas son identificadores de sección seguidos por el signo dos puntos, para ser utilizada la declaración goto.
2. Las declaraciones de expresiones son instrucciones que finalizan con el punto y coma.
3. La declaración condicional if, con la sintaxis if (condición) declaración else declaración. 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.

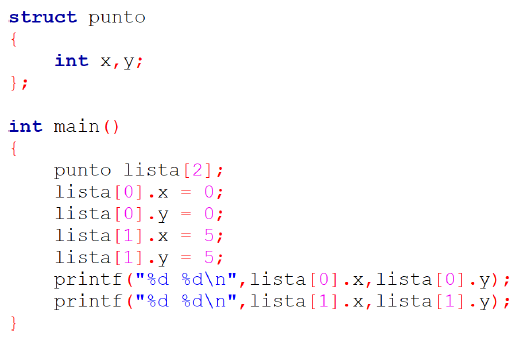
### Funciones

* 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



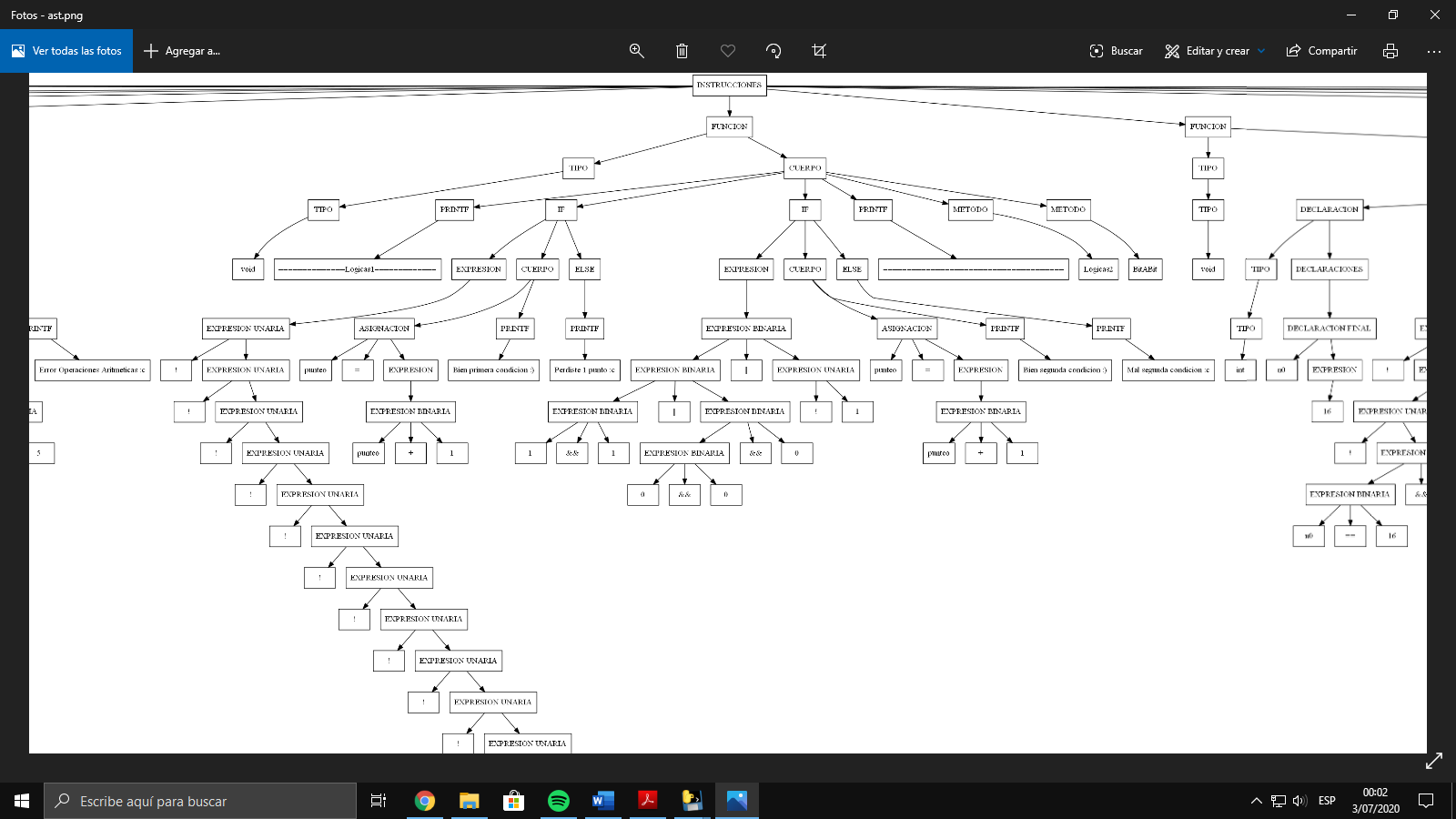




## Reportes generados

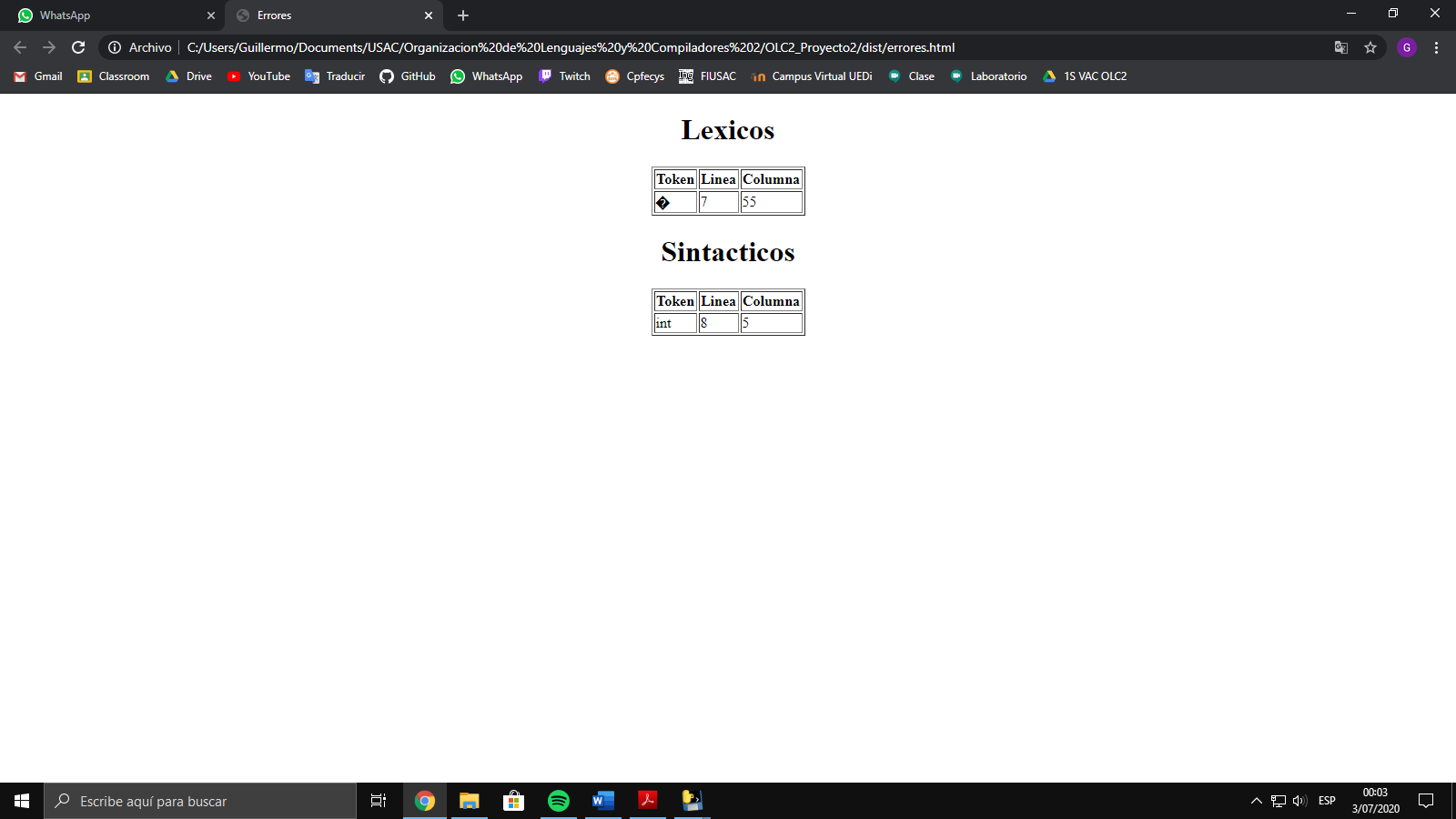
### AST

Contiene la forma del árbol sintáctico generado:



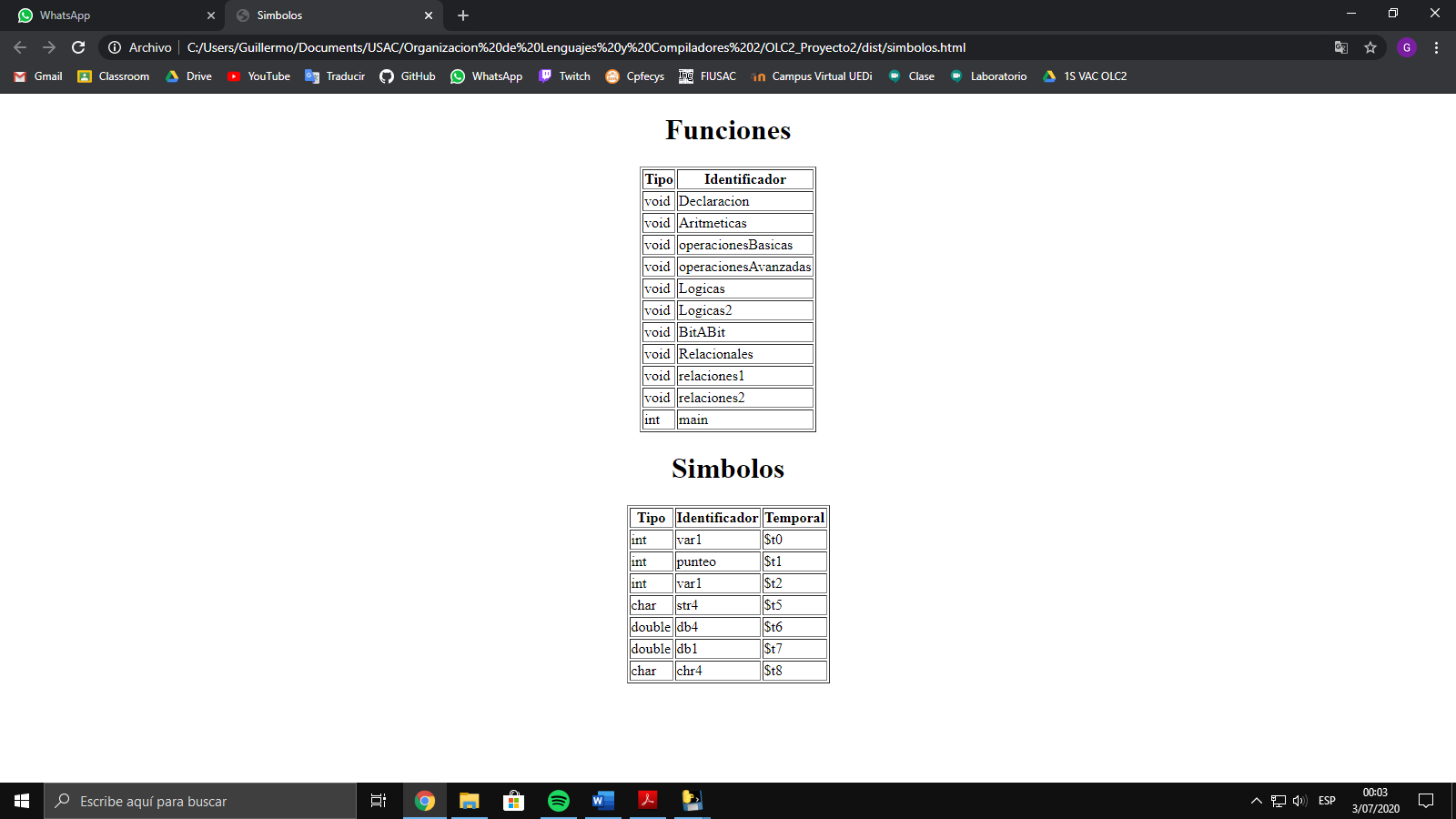
### Errores

Muestra los errores léxicos y sintácticos encontrados:



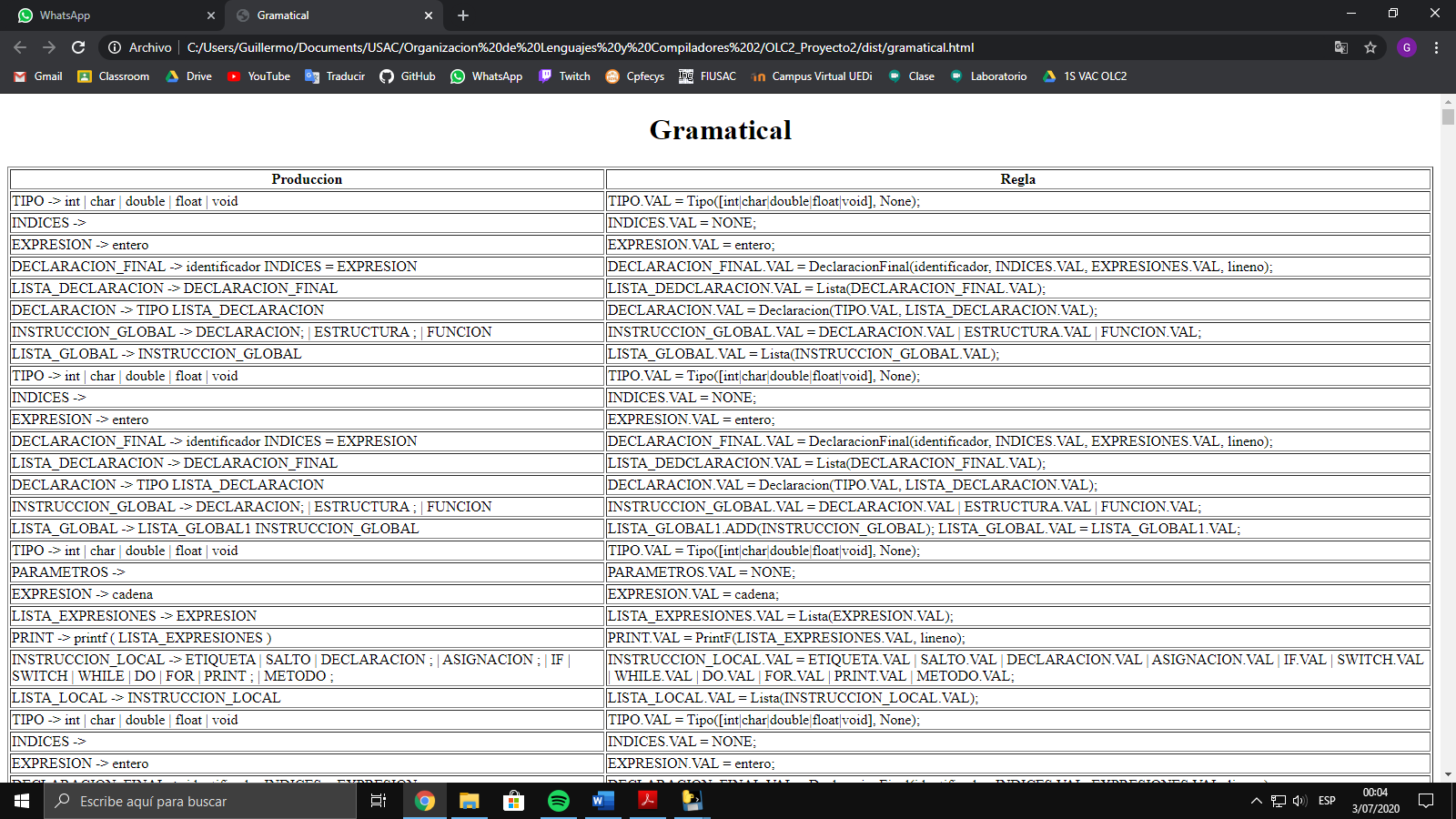
### Símbolos

Muestra las funciones y símbolos del archivo actual:



### Gramatical

Muestra la estructura gramatical del código fuente:



### Optimizaciones

Muestra las optimizaciones realizadas en el código de tres direcciones:

