UNIVERSIDAD TECNOLÓGICA DE SANTIAGO, UTESA

SISTEMA CORPORATIVO
FACULTAD DE INGENIERIA Y ARQUITECTURA
CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES

ASIGNATURA:

Compiladores
INF-920-001

TAREA & PRESENTACION SEMANA 7:

Compilador Pequeño

PRESENTADO A:

Iván Mendoza

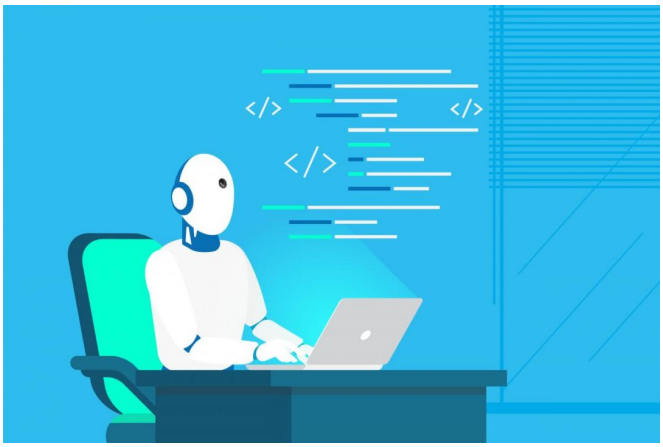
PRESENTADO POR:

Liván Herrera (2-16-0686)

Santiago de los Caballeros
República Dominicana
Marzo, 2022

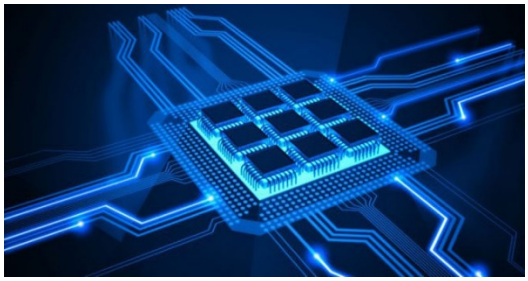
Investigar

- **Entorno de Ejecución**



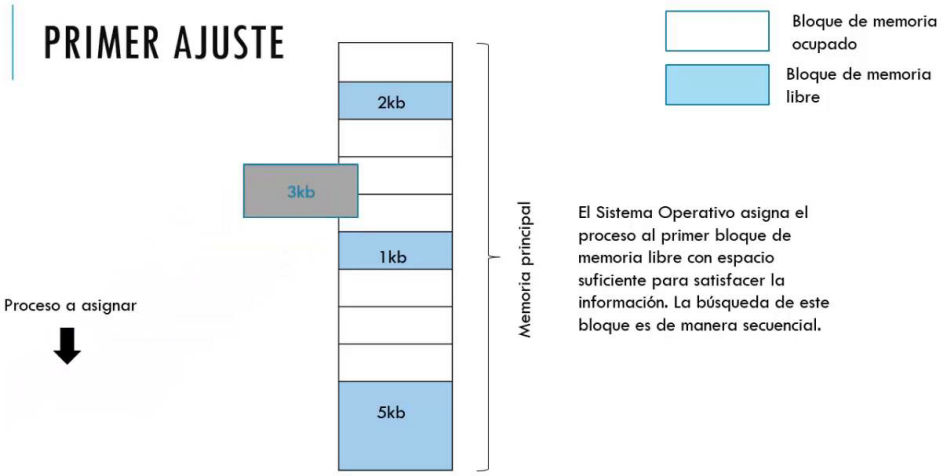
A la hora de ejecutar un programa en un ordenador el compilador no tiene control sobre los recursos de Hardware, dado que el sistema operativo o la máquina virtual en el caso de algunos lenguajes interpretados como Java, tienen el monopolio sobre estos recursos por ello los compiladores deben usar diferentes API y métodos provistos por el sistema operativo el cual también se encarga de asignarle la memoria a usar la cual deberá compartir con el resto de los programas, a todo este entorno y los factores que intervienen es lo que se conoce como el entorno de ejecución.

- **Organización de la memoria en tiempo de ejecución.**



La memoria en su tiempo de ejecución se va asignando a tareas en específico, la asignación de la memoria se lleva a cabo a través del sistema operativo. La organización o asignación de la memoria en tiempo de ejecución va desde los datos, el código, las librerías locales, compartidas, mapeo de archivos, heap y stack.

- **Estrategias de asignación de memoria.**



  - ✓ Por lo general, se divide la memoria en dos particiones, sistema operativo y procesos de usuarios.
  - ✓ Es necesario que exista una protección entre el sistema operativo y procesos de usuarios.
  - ✓ La verificación de acceso a la memoria se hace con registro de ubicación y límite.
  - ✓ Los métodos más comunes son mapas de bits y lista encadenada.

También se cuenta con dos tipos:

1. Asignación de memoria estática: aquí al programa se le asigna memoria en el momento de la compilación.
2. Asignación dinámica de memoria: aquí los programas se asignan con memoria en tiempo de ejecución.

- **Acceso a variables locales, no locales y globales.**

  - ✓ Las variables locales solo se pueden acceder dentro de la función en la fue creada.
  - ✓ Las variables no locales bien se podrían referir a variables globales, solo que estas se usan en funciones anidadas o anónimas, cuando la variable no puede ser ni local ni global, son accedidas en este mismo contexto.
  - ✓ Las variables globales son aquellas que se pueden acceder en cualquier ámbito del código.

- **Paso de parámetros.**

```
.....
    resultado = invertir(num);
.....

int invertir(int num)
{
    int inverso = 0, cifra;
    while (num != 0)
    {
        cifra = num % 10;
        inverso = inverso * 10 + cifra;
        num = num / 10;
    }
    return inverso;
}
```

Existen 3 diferentes tipos de paso de parámetros, los cuales son: paso por valor, paso por referencia y paso por valor resultado.

✓ El paso por valor consiste en copiar contenido y almacenarlo en otra dirección de memoria.

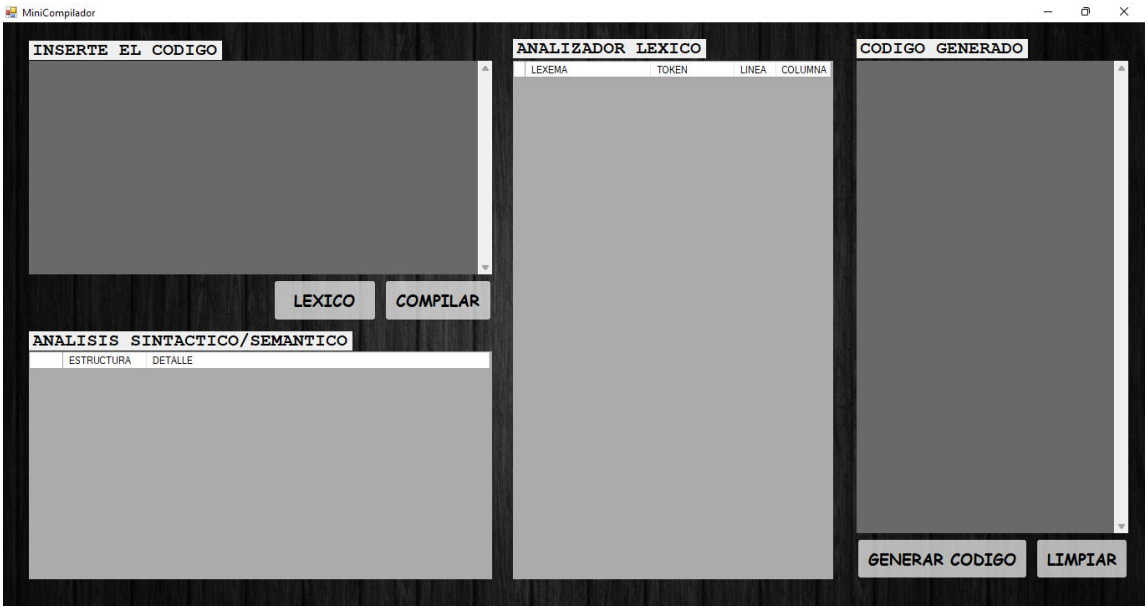
  - ✓ El paso por referencia, este consiste en pasar la dirección de memoria donde se encuentra el valor, en lugar del valor como tal.
  - ✓ El paso por valor resultado el cual consiste en que los valores funcionan como si se pasaran de argumentos a una función y al final se le copiaran los valores resultados

Ejercicios y Presentación:

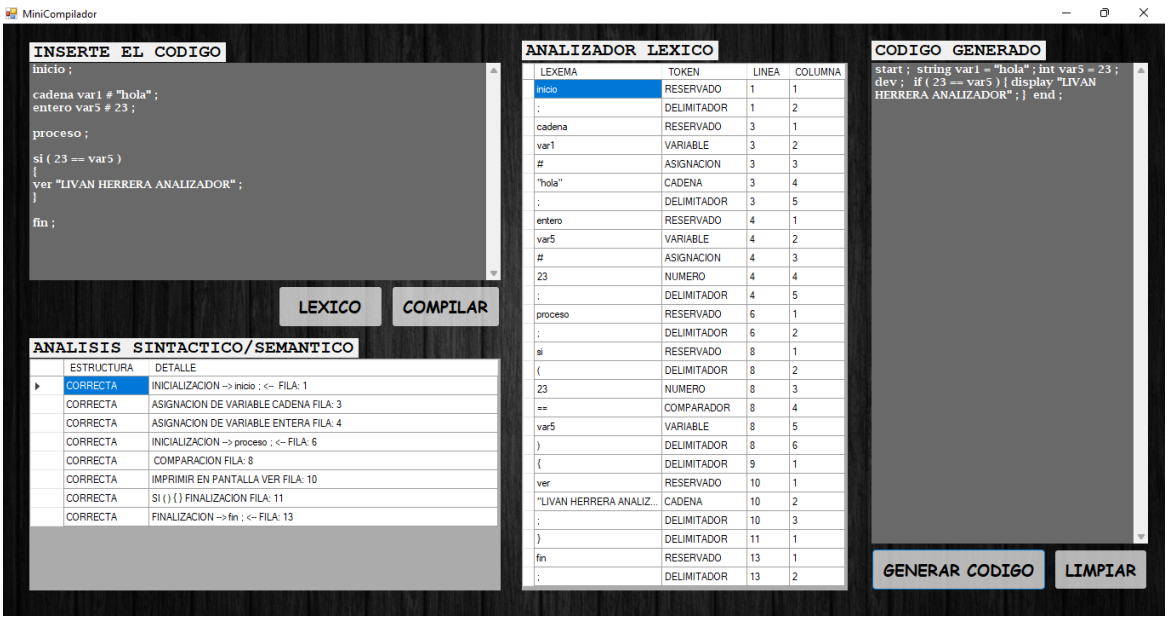
1. **Crear un pequeño Compilador con las siguientes características:**

   a) **Analizador Léxico.** OK
   b) **Analizador Sintáctico.** OK
   c) **Analizador Semántico.** OK
   d) **Generar Tabla de Símbolos.** OK
   e) **Generador de Código Intermedio.** OK
   f) **Lenguaje de programación a traducir el código del compilador.** OK
   g) **(Utilizar cualquier herramienta) en el Lenguaje de programación de su preferencia.** OK

2. **Documentar su Compilador y lenguaje de programación para realizar las pruebas.**



La interfaz del compilador es como se muestra en la imagen, el cual cumple con las funciones básicas de analizar el código de modo léxico, semántico y sintáctico; del mismo modo la generación del código a un nuevo lenguaje.



Aquí tenemos un código el cual funciona correctamente al 100%, es decir que no muestra errores, hace todo el análisis y la generación del nuevo código, más adelante se mostrarán algunas con errores para que se vea el funcionamiento correcto de cada analizador. El código usado es el siguiente:

```
inicio ;

cadena var1 # "hola" ;
entero var5 # 23 ;

proceso ;

si ( 23 == var5 )
{
ver "LIVAN HERRERA ANALIZADOR" ;
}

fin ;
```

Cuenta con 4 botones:

-LEXICO: Tal como su nombre lo indica, ejecuta la parte del analizador léxico.

-COMPILAR: Ejecuta tanto la parte sintáctica, como la semántica.

-GENERAR CODIGO: Se encarga de traducir el código a un nuevo lenguaje.

-LIMPIAR: Limpia toda la pantalla para así poder probar con un nuevo código.

Aquí se muestran dos errores léxicos, los cuales son el cierre con punto y coma y la forma incorrecta de asignación. El código utilizado:

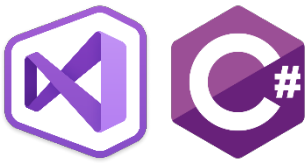
```
inicio


cadena var2 # "Hola" ;

entero var3 # 5 ;


fin ;
```

Como error sintáctico se pueden ver los dos últimos los cuales debían recibir un asignador en este caso el signo # y también cerrar o finalizar el código con punto y coma ;
Luego como error semántico la segunda línea, en la cual se tiene una variable declarada como entero y recibe una cadena. El código utilizado:

```
inicio ;

entero var1 # "Hola" ;
entero var2 # 6 ;
entero var3 # 5 ;
entero var4 ;

var4 == var2 + var3 ;

fin
```

Para la generación del código nuevo cambia las palabras reservadas a otras similares a las usadas en C#, por ejemplo, si cambia a if, mientras a while y así sucesivamente como se muestra en las imágenes.

```
public void Generacion_Codigo()
{
    string[] reservado = { "inicio", "proceso", "fin", "si", "ver", "mientras", "entero", "cadena" };
    string[] reservado2 = { "start", "dev", "end", "if", "display", "while", "int", "string" };
```



La herramienta y lenguaje utilizado fueron: Visual Studio y C#.

3. **Subir el código a GitHub, enviar el código fuente y el ejecutable del proyecto.**

https://github.com/livanh1/MiniCompilador.git