**UNIVERSIDAD CENTROAMERICANA**

**“JOSÉ SIMEÓN CAÑAS”**

****

**PROYECTO FINAL**

**ANALIZADOR SINTÁCTICO Y MANEJO DE ERRORES**

**FACULTAD DE INGENIERÍA Y ARQUITECTURA**

**TEORÍA DE LENGUAJES DE PROGRAMACIÓN**

**CICLO 02-2021**

**INTEGRANTES:**

Bryan Francisco Ceren Brito 00139817

Julio Edgardo Flores Giron 00081817

Alvaro García García Cornejo 00111317

Francisco Josue Molina Lopez 00300917

Gerardo Andrés Castro Lemus 00032515

**NOVIEMBRE DE 2021**

**Definimos la gramática a utilizar:**

La gramática fue diseñada para un subconjunto del lenguaje C, donde se incluye, declaración de variables, tipos de datos, instrucciones condicionales, manejo de comentarios, manejo de arreglos, cadenas, palabras claves y operadores aritméticos

| 1 | **S** -> int main ( ) { **S1** return constante\_int ; } |
| --- | --- |
| 2 | **S** -> ξ |
| 3 | **S1** -> **I S1** |
| 4 | **S1** -> **S2 S1** |
| 5 | **S1** -> **Z S1** |
| 6 | **S1** -> ξ |
| 7 | **I** -> if ( **C** ) { **S1** } **J** |
| 8 | **J** -> else { **S1** } |
| 9 | **J** -> ξ |
| 10 | **C** -> **B D B G** |
| 11 | **D** -> < | > | == | != | && |
| 12 | **B** -> identificador | constante\_int |
| 13 | **G** -> **D B G** |
| 14 | **G** -> ξ |
| 15 | **S2** -> **T** identificador **R** |
| 16 | **R** -> , identificador **R** |
| 17 | **R** -> [ constante\_int ] **N** |
| 18 | **R** -> = **P** |
| 19 | **R** -> ; |
| 20 | **P** -> **L H** |
| 21 | **P** -> ' identificador ' **K** |
| 22 | **H** -> , identificador **R** |
| 23 | **H** -> ; |
| 24 | **K** -> , identificador **R** |
| 25 | **K** -> ; |
| 26 | **N** -> , identificador **R** |
| 27 | **N** -> ; |
| 28 | **N** -> = { constante\_int **U** |
| 29 | **U** -> , constante\_int **U** |
| 30 | **U** -> } **V** |
| 31 | **V** -> , identificador **R** |
| 32 | **V** -> ; |
| 33 | **L** -> constante\_int | constante\_float |
| 34 | **T** -> int | float | char |
| 35 | **Z** -> identificador = **E** ; |
| 36 | **E** -> **Q A** |
| 37 | **A** -> + **Q A** |
| 38 | **A** -> - **Q A** |
| 39 | **A** -> ξ |
| 40 | **Q** -> **F1 W** |
| 41 | **W** -> \* **F1 W** |
| 42 | **W** -> / **F1 W** |
| 43 | **W** -> % **F1 W** |
| 44 | **W** -> ξ |
| 45 | **F1** -> ( **E** ) |
| 46 | **F1** -> identificador |
| 47 | **F1** -> constante\_int |
| 48 | **F1** -> constante\_float |

**Listamos los terminales**

| **N°** | **Terminales** |
| --- | --- |
| 1 | int |
| 2 | main |
| 3 | ( |
| 4 | ) |
| 5 | { |
| 6 | return |
| 7 | constante\_int |
| 8 | ; |
| 9 | , |
| 10 | ' |
| 11 | < |
| 12 | > |
| 13 | = |
| 14 | + |
| 15 | - |
| 16 | \* |
| 17 | / |
| 18 | % |
| 19 | } |
| 20 | ξ |
| 21 | if |
| 22 | else |
| 23 | identificador |
| 24 | | |
| 25 | float |
| 26 | char |
| 27 | constante\_float |

**Listamos los no terminales**

| **N°** | **No Terminales** |
| --- | --- |
| 1 | **S** |
| 2 | **S1** |
| 3 | **S2** |
| 4 | **I** |
| 5 | **Z** |
| 6 | **C** |
| 7 | **J** |
| 8 | **B** |
| 9 | **D** |
| 10 | **G** |
| 11 | **T** |
| 12 | **R** |
| 13 | **N** |
| 14 | **P** |
| 15 | **L** |
| 16 | **H** |
| 17 | **K** |
| 18 | **U** |
| 19 | **V** |
| 20 | **E** |
| 21 | **Q** |
| 22 | **A** |
| 23 | **F1** |
| 24 | **W** |

**Obtenemos los first y los follows**

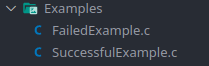
Para poder generar nuestra tabla de análisis sintáctico sacamos los first y los follow de nuestros no terminales.

| **No Terminales** | **FIRST** | **FOLLOW** |
| --- | --- | --- |
| **S** | { int, ξ } | { $ } |
| **S1** | { if, int, float, char, identificador, ξ} | { return , $} |
| **S2** | { int, float, char} | { if , identificador, $ } |
| **I** | { if } | { if , int, float, char , identificador , $} |
| **Z** | { identificador } | { if , int, float, char , identificador, $} |
| **C** | { identificador, constante\_int } | { ) , $ } |
| **J** | { else, ξ } | { $ } |
| **B** | { identificador, constante\_int } | {< , >, ==, !=, &&, $ } |
| **D** | {< , >, ==, !=, && } | { identificador, constante\_int } |
| **G** | {< , >, ==, !=, &&, ξ } | { $ } |
| **T** | { int, float, char} | { identificador, $} |
| **R** | { identificador, [ , = , **;** } | { $ } |
| **N** | { **,** , **;** , = } | { $ } |
| **P** | { constante\_int , ' } | { $ } |
| **L** | { constante\_int, constante\_float } | { **,** , **;** $ } |
| **H** | { **,** , **;** } | { $ } |
| **K** | { **,** , **;** } | { $ } |
| **U** | { **,** , **}** } | { $ } |
| **V** | { **,** , **;** } | { $ } |
| **E** | { ( , identificador, constante\_int, constante\_float} | { **;** , ) $} |
| **Q** | { ( , identificador, constante\_int, constante\_float} | { + , - $} |
| **A** | { + , -, ξ } | { $ } |
| **F1** | { ( , identificador, constante\_int, constante\_float} | { \* , / , % , $ } |
| **W** | { \* , / , %, ξ } | { $ } |

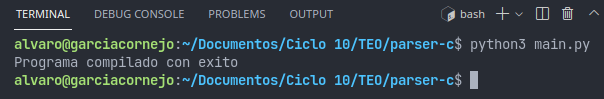
Luego procedemos a construir nuestra tabla de análisis sintáctico, y a generar el código de nuestro parser. (Ver en el Excel adjunto).

El analizador sintáctico, fue escrito en python, donde tenemos nuestro main.py donde está toda la lógica del análisis sintáctico.

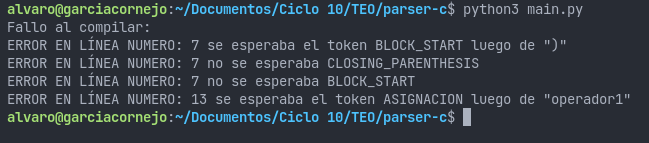
Contamos con 2 ejemplos de prueba, uno donde no existen errores de sintaxis donde esperamos que sea compilado exitosamente, y el otro ejemplo en el cual se presentan algunos errores de sintaxis.



La salida esperada para cuando el proceso es exitoso es la siguiente:



La salida esperada para cuando se encuentra un error de sintaxis:



**Referencias:**

* A Grammar for the C- Programming Language. Sitio web: <http://marvin.cs.uidaho.edu/Teaching/CS445/c-Grammar.pdf>