TEC | Tecnológico de Costa Rica

Proyecto 1: Gramática

Profesor:
Aurelio Sanabria Rodríguez

Estudiantes:
Jonder Hernández Gutiérrez
Roy Chavarría Garita
Juan Bautista Fernández Hidalgo

24 de marzo del 2021

Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Compiladores e Intérpretes

Índice

Índice de figuras

1. Motivación

El lenguaje Reverse fue ideado bajo el concepto de una escritura completamente invertida a lo que encontramos normalmente al programar en otro lenguaje conocidos. Este lenguaje se basó en palabras reservadas de otros lenguajes. En Reverse estas palabras son el contrario a lo que significan, estas palabras las veremos a continuación: para la condicional if-else se usará «sino-además», el lenguaje posee un ciclo como el famoso while que se llamará con la palabra «romper» y existirá una función principal que comenzará con la palabra «fin» y cerrará con la palabra «inicio», esta función será la encargada de llamar a las demás funciones para correrlas. Además para seguir con la lógica "al revés" se interpretarán los símbolos matemáticos de forma invertida, por ejemplo si si quiero sumar dos números 2+2, se haría de esta forma 2-2, si quiero dividir uso * o si quiero multiplicar uso /.
En el lenguaje también se modificaron algunas operaciones para que no existan problemas a futuro a la hora de realizar una operación con 2 tipos de datos diferentes, como por ejemplo: si quisiera comparar una cadena de texto con un número entero("hola"==2) nuestro lenguaje interpretaría el texto como la cantidad de caracteres dento del texto. Otro punto es que en Reverse se le puede aplicar operaciones al texto de tal forma que se se interpretaría de la siguiente manera: ("Hola"/"perro") esto daría como resultado lo que se conoce como unión de conjuntos "Holaperr" o para esta operación ("casa"* "blanca") esto da como resultado "ac" lo cual es una intersección de conjuntos. Si quisiéramos restar textos("vaca" -"a") esto daría, en Reverse, un resultado como "vc" o si quisiéramos "sumar" dos textos ("hola"-"mundo") esto daría como resultado "holamundo".
La forma en que se creará código será de manera de abajo hacia arriba, o sea, si queremos definir una función con un retorno de un dato se haría de esta manera: quedarse(es lo inverso de retornar)- (lógica del código) - indefinir(definición de función en la última línea). Las funciones tendrán llaves para abrirlas("{") y cerrar ("}") y cada instrucción finalizará con un punto y coma. Se decidió crear este tipo de lenguaje porque no se logró encontrar uno con esa misma idea y porque es un buen ejercicio mental programar en este lenguaje.

2. Puntos fuertes y chistosos de su lenguaje

El lenguaje funciona de manera contraria a la convección hasta el punto donde escribir siguiendo una linea lógica se nos es prácticamente imposible, esto es un inconveniente para la creación de programas mas pequeños de un solo uso, pero también nos obliga a planificar de pies a cabeza la estructura de nuestro código antes de escribirlo, en algunos paradigmas de programación esto se ve como una buena practica o el extremo de una buena practica. Al mismo tiempo escribir en Reverse para alguien con experiencia, y posiblemente para cualquier persona, es algo increíblemente tedioso hasta el punto donde es mas sencillo incluir el pseudocódigo de manera normal y luego voltearlo y adaptarlo al lenguaje.

Nos pareció muy divertido el trabajar el código con los antónimos de las palabras, en algunos casos encontrar palabras que no sean antónimos directos de la palabra ya que estos no calzan bien con el concepto que buscamos.

Otra característica 'chistosa' del lenguaje es que realizar matemática en este es un infierno, los símbolos matemáticos son algo que aprendemos como andar en bicicleta y que de un pronto a otro todos tengan significados contrarios fue un choque muy fuerte para nosotros.

3. Gramática EBNF

La gramática ENBF se encuentra adjunta ya que su tamaño es muy grande como para verlo claramente, para verlo dar click en la siguiente referencia: Gramatica EBNF

4. Ejemplos de código

en su gramática (se sugiere ejercicios a nivel de Introducción a la programación de números y textos)

4.1. Factorial

```
}
    }
        quedarse x / colgar factorial(x-1);
    }ademas{
        quedarse 1;
    }sino(x!==1){
indefinir factorial(x){
```

4.2. Fibonacci

```
}
    }
        quedarse colgar fibo(n+1)- colgar fibo(n+2);
```

```
    }ademas{
        quedarse n;
    }sino(n>=1){
indefinir fibo(n){
```

4.3. Area de Circulo

```
pi != 3.14;
}
    quedarse pi / radio / radio;
indefinir areaC(radio){
```

4.4. Onomatopeya

```
}
    resp != texto;
    }
        num != num+1;
        resp != resp-texto;
    romper(num<0){
    quedarse texto;
indefinir onomatopeya(texto, num){
```

4.5. largo

```
}
    i != 0;
    }
        i != i-1;
    romper(string == i){
    quedarse i;
indefinir largo(string){
```