

Diseño de Compiladores

**MATRUSHKA**

Jose Luis Loya Cabrera A01111663

1. de Mayo, 2016

**Descripción del Proyecto**

**Visión y propósito del proyecto**

La visión de matrushka es poder encriptar y desencriptar programas enteros. Actualmente personas que quieren comunicarse de manera segura necesitan saber del tema y saber utilizar las herramientas. Con matrushka, la barra de dificultad disminuye. El usuario que recibe o ejecuta el programa solo tiene que introducir la contraseña para poder ejecutarlo, no ocupa conocimientos especiales.

**Objetivos del Proyecto**

Mi objetivo es llevar la encriptación a más lugares. Me fue imperativo dado la época que vivimos de vigilancia masiva. Otro objetivo es que fuera sencillo de utilizar. No quiero hacer la encriptación más complicada, si no más accesible.

**Alcance del Proyecto**

El proyecto fue diseñado para que la configuración del cifrado fuese sencilla. Es lo primero que vez cuando vez el código fuente. El lenguaje funciona con o sin cifrado. Tiene las funcionalidades que esperarías de un lenguaje educativo.

**Análisis de Requerimiento**

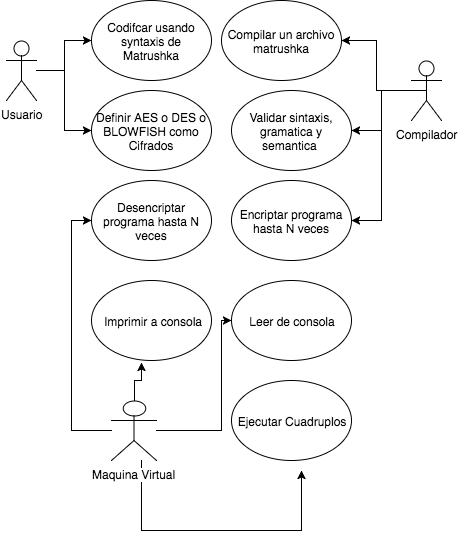
Requerimientos Funcionales:

* El lenguaje debe soportar los algoritmos de cifrado DES, AES y BLOWFISH
* El compilador deberá poder compilar código escrito con gramática valida.
* El compilador deberá detenerse al detectar errores.
* El compilador deberá cifrar el programa de 0 a N veces. (N siendo la cantidad de cifrados especificados).
* El lenguaje debe soportar los tipos entero, flotante, booleanas y string.
* El lenguaje debe soportar variables, vectores y matrices
* El lenguaje debe soportar condicionales if/else y ciclos while
* El lenguaje debe soportar funciones y parámetros
* El lenguaje debe soportar variables locales y globales
* El lenguaje debe soportar scope de variables
* El lenguaje debe soportar operaciones de concatenación en strings
* El lenguaje debe soportar lectura de stdin
* El lenguaje debe soportar escritura a stdout

Requerimientos No Funcionales:

* La gramática deberá ser sencilla
* La máquina virtual y el compilador deberán utilizar OpenSSL para cifrados
* El compilador deberá estar codificado en ANSI C
* La máquina virtual deberá estar codificada en Python3

**Casos de uso**



**Descripción de las pruebas**

Los archivos dentro de “/tests/” consideran las siguientes cosas:

* Prueba de ciclos y condiciones
* Prueba de llamadas a funciones
* Prueba de expresiones aritméticas
* Prueba de expresiones con strings
* Prueba de variables, vectores y matrices
* Prueba de compilación y ejecución con 0 cifrados
* Prueba de compilación y ejecución con 1 cifrado
* Prueba de compilación y ejecución con 3 cifrados

Descripción del Proceso

|  |  |
| --- | --- |
| **Fecha** | **Comentario** |
| Thu May 4 | Codigo ya limpio y con comentarios |
| Thu May 4 | Cleaned about 80% of the code |
| Thu May 4 | Cleaned about 40% of the code |
| Thu May 4 | Cleaned about 20% of the code |
| Tue May 3 | Removed old initialization script |
| Tue May 3 | Moved source files. Created Makefile |
| Tue May 3 | Encryption works |
| Tue May 2 | Ya funciones vectores y matrices |
| Tue May 2 | Máquina virtual lista. Arreglos masomenos |
| Mon May 1 | Added era\_size reference at bottom |
| Mon May 1 | New standard for opcodes. Deleted deadcode. |
| Sun Apr 30 | Code generation for CYCLES READY |
| Sun Apr 30 | Code generation for condicionales |
| Sun Apr 30 | Generacion de codigo secuencial listo |
| Sat Apr 29 | IR Code for simple if's |
| Sat Apr 29 | Added memory addressing to symbol table |
| Sat Apr 29 | Do semantics analysis and code generation in just one attribute for expressions |
| Sat Apr 29 | Cleaned up even more. Better naming and structure |
| Sat Apr 29 | Cleaning up work: RENAMING, DELETING DAEDCODE, ADDING COMMENTS |
| Fri Apr 28 | Generacion de codigo sencillo para expresiones |
| Fri Apr 28 | Semantica de Variables y Expresiones Terminada |
| Fri Apr 28 | Switched 1era y 2da entrega folder. They were in the wrong order |
| Thu Apr 27 | Entrega 2 terminada |
| Thu Apr 27 | Trabajando en la segunda entrega |
| Tue Apr 25 | Primera entrega lista |

Fue el proyecto más largo, complicado y gratificante que hice en mi carrera. Combine mis dos lenguajes favoritos para programarlo y también de inspiración. El hecho de que el lenguaje tenía que ser temático, me hizo escoger algo que valiera la pena. La verdad lo empecé muy tarde porque trabajo y estudio y hubo varias veces que pensé en tirar la toalla. Sin embargo, lo termine y esto me da mucho orgullo.

.

Jose Luis Loya Cabrera

**Descripción del Lenguaje**

**Nombre**

El lenguaje se llama Matrushka. Está inspirado en la muñeca rusa que al abrirla tiene otra muñeca que al abrirla trae otra muñeca y así recursivamente hasta llegar al final. De ahí nace la inspiración. Los programas de este lenguaje pueden cifrarse múltiples veces y para ejecutarlo tienes que descifrarlo múltiples veces.

Descripción

El lenguaje le permite al usuario definir un cifrado y una clave para descifrarlo. Ese es el feature principal. La sintaxis está inspirada en C y Python. Dándole esa vibra de simplicidad y seriedad al mismo tiempo. El lenguaje le permite al usuario usar variables, vectores, matrices, funciones, whiles, condiciones IFs, lectura y escritura en consola. En cuestión de tipos solo son 4: int, double, bol y string.

Errores que pueden ocurrir

El compilador detecta y avisa de los siguientes errores:

* Uso de variables o funciones previamente no declaradas
* Doble declaración de variables con el mismo nombre
* Usos no permitidos de operandos en ciertos operadores
* Regresar un dato distinto al de la función
* Usar un algoritmo de cifrado distinto a “AES”, “DES” o “BLOWFISH”
* Expresiones distintas a booleanas en condicionales y ciclos
* Errores de gramática

La máquina virtual avisa sobre los siguientes errores:

* La clave de cifrado es incorrecta
* Cuando detecta un cuádruplo no registrado

**Descripción del Compilador**

Descripcion del compilador

Equipo de computado, lenguaje y librerías usadas

Descripcion del Analisis de Lexico

Los regex y los tokens

Gramatica (Sin código)

Descripcion de la generación de código

Descripcion de la administración de memoria

Osea todas las estructuras que jalamos

Descripcion de la Maquina Virtual

Equipo de computo, lenguaje y utilerías especiales usadas

Descripcion detallada del proceso de Administracion de memoria

Estructguras y mapeo de virtuales a reales

Pruebas del funcionamiento del lenguaje

Listaods documentados del proyecto

Manual de Usuarios “Quick reference Manual para programadores”