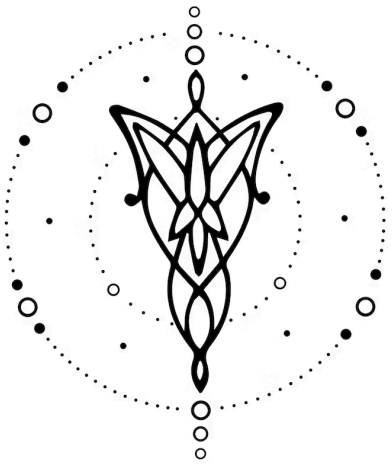

Especificación

Diseño e Implementación de un Lenguaje

v1.0.0

1. Equipo

Nombre	Apellido	Legajo	E-mail
Jonathan Lucas	Blankleder	64660	jblankleder@itba.edu.ar
Agustin German	Ramirez Donoso	64716	aramirezdonoso@itba.edu.ar

2. Repositorio

La solución y su documentación serán versionadas en: github

3. Dominio

Desarrollar un lenguaje que permita pasar de código C a un texto (string) que sea un JSON comprimido con zlib y luego codificado con Base64 el cual pueda ser importado al juego Factorio como un plano, para luego ser ejecutado dentro del juego.

Específicamente, el proyecto se basará en un parser de C a un pseudo C, el cual será compilado a código Assembler, ese Assembler a un JSON, y finalmente el archivo .JSON será comprimido y codificado. Todo el proceso se podrá ejecutar por pasos para ver cada archivo intermedio.

El lenguaje sólo podrá ser ejecutado en Factorio aunque se proveerá un instructivo de instalación y ejecución para casos de testeo. Además se proveerán videos para demostrar el funcionamiento del compilador.

Aún así, va a ser posible ver el código assembler y se proveerá un compilador aparte de "pseudo-assembler" a assembler para realizar tests y comprobar el funcionamiento correcto del compilador. Aún así, de esta forma solo se podrán enviar variables al momento de ejecutar el programa y los resultados deberán ser vistos en los registros de assembler

La finalización de estos lenguajes y compiladores permitirá la facilitación de escribir código dentro de una computadora en factorio, permitiendo asi romper los límites antes creídos imposibles dentro del juego

4. Construcciones

El lenguaje desarrollado debería ofrecer las siguientes construcciones, prestaciones y funcionalidades:

I. Se podrán ejecutar funciones dentro de la función main
II. Se proveerá parte de la funcionalidad de la librería Math.h
III. El lenguaje a escribir se podrá compilar con GCC
IV. Las variables podrán ser de tipo int, boolean o point
V. Se proveerá una implementación básica de punteros
VI. Se proveerán las funciones drawPoint(), drawLine(), drawNum()
VII. Se proveerá una función que verifique que el input del usuario sea válido para guardar en una variable

5. Casos de Prueba

Se proponen los siguientes casos iniciales de prueba de **aceptación**:

I. Una calculadora con operaciones +,- y *
II. Un programa que dibuje un punto en una posición a eleccion de la pantalla
III. Un programa que quiera ejecutar código assembler

IV. Un programa que envie parámetros a main
V. Un programa que reciba input del usuario cuando vaya a ser ejecutado
VI. Un programa que reciba input del usuario durante la ejecución del programa
VII. Un programa en el cual su función main llama a otra función dentro del programa
VIII. Un programa que mire los valores de los registros
IX. Un programa que utilice memoria RAM
X. Un programa con punteros dentro

Además, los siguientes casos de prueba de **rechazo**:

I. Un programa no compilable con GCC
II. Un programa sin una función de entrada main
III. Un programa que utilice strings
IV. Un programa que busque utilizar memoria no existente
V. Un programa que intente escribir un número de más de un digito con drawNum()
VI. Un programa que intente incluir una librería (no es necesario incluir nuestra librería)
VII. Un programa que defina más de una variable por línea

6. Ejemplos

Calculadora básica:

```
// Muestra la suma de 2 números elegidos en ejecución en pantalla:
int main(int argc, *int argv){
    int num1;
    int num2;
    num1 = listen(); // la función listen espera el input del usuario
    num2 = listen();
    num1 = num1 + num2;
    drawNum(10,10,*num1);// dibuja un número en la posición 10,10 de la pantalla
}
```

Crear un pong estático:

```
struct Rec{
   int posX;
   int posY;
   int sizeX;
   int sizeY;
   point[4] drawable; // estos 4 puntos serán usados para dibujar el rectángulo cuando sea
necesario.
};
```

```
// Crea la paleta y devuelve el puntero a la paleta
*Rec createRec(*int posX, *int posY, *int sizeX, *int sizeY){
    point[4] Rec;
    Pad[0][0] = posX;
    int aux = *posX + *sizeX;
    Pad[0][1] = posY;
    ...
}

// Crea todas las variables (por separado) y llama a las funciones createPad
int main(int argc, *int argv){
    int xPad1;
    int xPad2;
    int yPad1;
    int yPad2;
    int xBall;
    int yBall;
    int sizePadX;
    int sizePadY;
    int sizeBall;
    xPad1 = 10;
    int xPad2 = 80;
    int yPad1 = 10;
    int yPad2 = 10;
    int xBall = 45;
    int yBall;
    Pad pad1;
    ...
    pad1 = createRec(...);
    pad2 = createRec(...);
    ball = createRec(...)
```