Especificación

Diseño e Implementación de un Lenguaje

v1.0.0

1. Equipo
2. Repositorio
3. Dominio
4. Construcciones
5. Casos de Prueba
6. Ejemplos

1. Equipo

Nombre	Apellido	Legajo	E-mail
Pedro	Seggiaro	63337	pseggiaro@itba.edu.ar
Santiago Joaquín	Nartallo Galvagno	62208	snartallogalvagno@itba.edu.ar
Martina	Schvartz Tallone	62560	mschvartztallone@itba.edu.ar
Candela Denise	Silva Diniz	63402	csilvadiniz@itba.edu.ar

2. Repositorio

La solución y su documentación serán versionadas en: GitHub

3. Dominio

El objetivo del proyecto es desarrollar un compilador con un enfoque visual y organizativo que genere árboles en estilo ASCII art mediante un lenguaje de entrada más complejo y funcional. El objetivo es transformar la herramienta en un sistema flexible que no solo genere árboles basados en parámetros básicos, sino que también permita una amplia gama de operaciones y composiciones.

El lenguaje permitirá especificar parámetros cómo altura, probabilidad de ramificación, densidad de hojas y curvatura de las ramas, pero también incluirá la capacidad de instanciar y componer múltiples árboles. Los usuarios podrán crear un bosque de árboles, definir variables (cómo por ejemplo **'tree'** y **'forest'**), y utilizar bucles y ecuaciones para generar árboles dinámicamente.

Se incorporarán operadores aritméticos, lógicos y relacionales para permitir comparaciones y cálculos entre árboles. Los usuarios podrán utilizar operadores como **t1 + t2** (donde ***t1*** y ***t2*** serían dos árboles diferentes) para combinar árboles o **t1 < t2** para comparar sus características. Además, se incluirán estructuras condicionales como *IF-THEN-ELSE* u opciones de *pattern-matching* para una mayor flexibilidad en la definición de los árboles.

Los usuarios podrán componer árboles y bosques mediante la definición de variables y el uso de bucles, permitiendo así, generar múltiples árboles a partir de una única ecuación o configuración, ofreciendo una mayor capacidad de personalización y variación de los resultados.

Con respecto al formato visual y aspecto de los árboles, cada usuario podrá personalizar los árboles mediante su propia elección de caracteres para representar las hojas (permitiéndole seleccionar **entre tres**), colores para los distintos árboles individuales o dentro de un bosque, entre otras opciones.

Por último, para no limitar la herramienta a solamente archivos de texto, ésta ofrecerá una salida en HTML que permitirá una visualización más interesante. Teniendo en cuenta los distintos formatos disponibles incluyendo colores, superposiciones en el caso de bosques y efectos de profundidad para darle perspectiva al mismo, se podrán ajustar aspectos visuales como el z-index para hacer de estas, unas representaciones más detalladas y atractivas.

Este enfoque amplio hará de la herramienta una muy versátil y que le otorgará al usuario libertad y opciones al momento de crear y generar los resultados.

4. Construcciones

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

(I). Generar el ascii art de un árbol.
(II). Determinar la altura del árbol a generar, dónde cero implica que se ramifica desde la base.
(III). Determinar entre 3 símbolos, '#' '$' '&', para utilizar como hojas del árbol.
(IV). Determinar el porcentaje de ramas del tronco del árbol, como un número entre 0 y 10.
(V). Determinar hasta 3 colores para utilizar en el resultado.
(VI). Determinar la densidad del follaje del árbol como un número entre 0 y 10.
(VII). Indicar un mensaje a incluir en el resultado.
(VIII). Generar el ascii art de un bosque.

(IX). Sumar modelos de árboles entre sí, generando un bosque con variedad de diseños.
(X). Añadir árboles a un bosque.
(XI). Comparar árboles por altura y/o densidad de follaje.
(XII). Copiar parámetros de un árbol a otro.

5. Casos de Prueba

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

(I). Un programa que dibuje un árbol predeterminado(sin parámetros).
(II). Un programa que dibuje un bosque predeterminado(sin parámetros)
(III). Un programa que dibuja un árbol de altura 6 con 3 de ramas.
(IV). Un programa que dibuje un árbol compuesto del símbolo "#" y follaje "abundante".
(V). Un programa que, mediante un IF-ELSE, imprima en el resultado el árbol con mayor altura.
(VI). Un programa que instancie un árbol a partir de otro y luego modifique únicamente su parámetro de símbolo para hojas.
(VII). Un programa que dibuje un bosque a partir de 3 árboles, definidos y personalizados en todos sus parámetros previamente en el código. El bosque instanciado por medio de un loop.
(VIII). Un programa que genere 10 árboles de alturas alternas
(IX). Un programa que genere 1 árbol, luego un bosque y luego otro árbol.
(X). Un programa que genere un árbol de densidad máxima y altura mínima resultando en algo que se asemeja a un arbusto

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

(I). Un programa vacío.
(II). Un programa que define un bosque en base a un árbol inexistente.
(III). Un programa que asigne a un tipo "tree" un "forest".
(IV). Un programa que ejecute grow sobre un árbol inexistente.
(V). Un programa que no inicie con "plant" y/o no finalice con "chop".
(VI). Un programa que no separe sus sentencias por medio de ";".
(VII). Un programa que no indique el parámetro a inicializar dentro de los constructores.
(VIII). Un programa que quiera sumar un tipo de dato distinto de "tree" o "forest" a un forest.
(IX). Un programa que quiera instanciar el mismo parámetro múltiples veces dentro de un mismo constructor.
(X). Un programa que genere árboles con todas las características numéricas negativas.

6. Ejemplos

Programa que crea dos árboles de determinadas características y los une en una variable *forest*, lo cual representaría una variable de tipo *array*.

```
plant

    tree t1 = sprout();

    t1.height = 3;
    t1.density = 0;

    tree t2 = sprout(height = 6, branches = 1);
    t2.leaf = &;

    forest f1 = t1 + t2;

    grow(t1);
    grow(t2);

    grow(f1);

chop
```

Programa que genera un bosque de 10 árboles que tienen alturas crecientes

```
plant
    num h = 10;
    forest f1;

    for(num i = 0; i < 10; i++){
       tree t1 = sprout(height = h);
       t1.height = h;
       h = h + 5;
       f1 += t1;
    }

    grow(f1);
chop
```