WALL-E PAINTS

INFORME

Este proyecto es un compilador capaz de recibir instrucciones (dibujar rectas, círculos, rectángulos, entre otros), mediante el uso de variables, expresiones aritméticas y booleanas de gran complejidad, bucles GoTo que evalúan una expresión booleana y se dirigen hacia cierta etiqueta en el código. También consta de funciones para obtener información sobre el canvas actual o sobre la posición de Wall-E. Los colores disponibles están nombrados en la instrucción Color, y las direcciones del puntero en DrawLine.

El programa contiene en la escena principal:

- Una cuadrícula que puede tomar cualquier tamaño cuadrado desde 1 hasta 200 (para evitar entradas demasiado grandes) con el botón “Redimensión”.
- La entrada de código.
- El panel de errores.
- El botón de “Compilar” para restablecer y ejecutar el código de entrada.
- Botones de “Guardar” y “Cargar”, los cuales generan o utilizan archivos de extensión .pw.
- El botón de “Reglas”, que genera un panel que contiene toda la información necesaria para utilizar el programa. Estas reglas se muestran a continuación:

INSTRUCCIONES

Spawn(int x, int y)
Todo código debe comenzar con un comando Spawn(int x, int y) y solo puede utilizarse esta instrucción una vez. Este comando inicializa a Wall-E sobre el canvas. Las entradas **x, y** representan las coordenadas iniciales.

Color(string color)
Este comando cambia el color del pincel. Los colores a elegir son los siguientes:
”Red”
”Blue”
”Green”
”Yellow”
”Orange”
”Purple”
”Black”
”White”
”Transparent”
El color por defecto del pincel es ”Transparent”.

Size(int k)
Modifica el tamaño del pincel. El grosor debe ser un número impar, por lo que si la entrada es par, se utilizará antecesor. El tamaño por defecto del pincel es de un pixel.

DrawLine(int dirX, int dirY, int distance)
Dibuja una línea sobre el canvas que comienza en la posición actual de Wall-E y termina a distancia distance en píxeles en la dirección establecida. Las entradas válidas para dirX y dirY son: -1, 0, 1, dando como resultado 8 direcciones:
(-1, -1) ← Diagonal arriba izquierda.
(-1, 0) ← Izquierda
(-1, 1) ← Diagonal abajo izquierda.
(0, 1) ← Abajo
(1, 1) ← Diagonal abajo a la derecha.
(1, 0) ← Derecha.
(1, -1) ← Diagonal arriba derecha.
(0, -1) ← Arriba.
Después de esta instrucción, la nueva posición de Wall-E será en el final de la línea.

DrawCircle(int dirX, int dirY, int radius)
Dibuja un círculo de radio radius en la dirección establecida. La posición final de Wall-E será el centro del círculo. La línea del radio no se debe dibujar, solo la circunferencia.

DrawRectangle(int dirX, int dirY, int distance, int width, int height)
Similar al círculo pero dibujando un rectángulo. Wall-e se moverá en la dirección (dirX, dirY) una cantidad de píxeles igual a distance y esta posición representará el centro del rectángulo de largo width y altura height.

Fill()
Pinta con el color de brocha actual todos los píxeles del color de la posición actual que son alcanzables sin tener que caminar sobre algún otro color.

VARIABLES

Asignación de variables

Nuestro código de entrada también soporta asignación de variables numéricas o booleanas. La sintaxis de asignación es la siguiente:
Var <- *Expression*
Donde *var* tiene forma de cadena de texto formada por los 27 caracteres del alfabeto español, caracteres numéricos y el símbolo _. Además, no pueden comenzar con caracteres numéricos ni el símbolo _ y *Expression* es cualquier expresión aritmética o booleana. Una variable es numérica si su *Expression* es aritmética, y booleana si su *Expression* es booleana. Después de una asignación de variable debe haber un salto de línea.

Expresiones

Las expresiones pueden ser aritméticas o booleanas.

Expresiones aritméticas

Una expresión aritmética (o numérica) está formada por: Un literal: Numero entero.
Una variable numérica.
Una operación aritmética entre dos o más expresiones aritméticas.
Una invocación de función. ´
Las operaciones aritméticas que soporta el lenguaje son:

Suma (+)
Resta (-)
Multiplicación (*)
División (/)
Potencia (**)
Modulo (%)

Expresiones booleanas
Una expresión booleana puede evaluar como **Verdadera (true)** o **Falsa (false)** y está formada por:
Concatenación **and** (&&) entre dos comparaciones.
Concatenación **or** (||) entre dos comparaciones.
Comparación (==, >=, <=, >, <) entre dos variables numéricas o literales.
La operación **or** tendrá siempre más precedencia que la operación **and**.

FUNCIONES

Las funciones retornan un valor numérico que se puede asignar a una variable. Reciben como entradas variables o literales (no otras funciones). Después de una función debe haber un salto de línea.

GetActualX()
Retorna el valor **X** de la posición actual de Wall-E.

GetActualY()
Retorna el valor **Y** de la posición actual de Wall-E.

GetCanvasSize()
Retorna tamaño largo y ancho del canvas. Para un canvas de $n \times n$ se retorna n.

GetColorCount (string color, int x1, int y1, int x2, int y2)
Retorna la cantidad de casillas del color pasado de parámetro que hay en el rectángulo formado por las posiciones x1,y1 (una esquina) y x2, y2 (la otra esquina). Si cualquiera de las esquinas cae fuera de las dimensiones del canvas, retorna 0.

IsBrushColor(string color)
Retorna 1 si el color de la brocha actual es el color pedido, 0 en caso contrario.

IsBrushSize(int size)
Retorna 1 si el tamaño de la brocha actual es size, 0 en caso contrario.

IsCanvasColor(string color, int vertical, int horizontal)
Retorna 1 si la casilla señalada está pintada del parámetro color, 0 en caso contrario. La casilla en cuestión se determina por la posición actual de Wall-E **(X, Y)** y se calcula como: (**X** + horizontal, **Y** + vertical). Note que si tanto horizontal como vertical son 0, se verifica entonces la casilla actual de Wall-E. Si la posición cae fuera del canvas retorna false.

SALTOS CONDICIONALES

Etiquetas

Una etiqueta es una cadena de texto que marca un lugar del código al cual se puede llegar a través de un GoTo. Las etiquetas no son asignaciones, instrucciones ni funciones, pues por sí mismas, no tienen efecto cuando llega su turno secuencial de ejecutarse. Una etiqueta, al igual que los nombres de variables, tiene forma de cadena de texto formada por los 27 caracteres del alfabeto, caracteres numéricos y el símbolo _. Además, no pueden comenzar con caracteres numéricos ni el símbolo _. Después de una etiqueta debe haber un salto de línea. Ejemplos de etiquetas válidas:
start_loop_1
this_is_where_my_loop_sadly_ends
my_programming_teachers_are_awesome

2.6.2. Saltos condicionales

Un salto condicional tiene la forma **GoTo** [**label**] (**condition**).

label debe ser una etiqueta declarada en el código tanto antes como después de la declaración del salto. Si se trata de hacer un salto a una etiqueta inexistente, debe desencadenarse un error de compilación.

condition es una variable booleana o una comparación (==, >=, <=, >, <) entre dos variables numéricas o literales. Los saltos condicionales tienen el efecto de que, si la condición tiene valor **Verdadero (true)**, el código continúe su ejecución en la línea de la **etiqueta** correspondiente. Si la condición tiene valor **Falso (false)**,

entonces la esta línea se ignora y se continua la ejecución con la línea siguiente. Después de un **GoTo** debe haber un salto de línea. Note que los paréntesis y corchetes son parte de la sintaxis del salto condicional.

EJEMPLO DE CÓDIGO

```
Spawn(0, 0)
Color("Black")
n <- 5
k <- 3 + 3 * 10
n <- k * 2
actual_x <- GetActualX()
i <- 0

loop_1
DrawLine(1, 0, 1)
i <- i + 1
is_brush_color_blue <- IsBrushColor("Blue")
Goto [loop_ends_here] (is_brush_color_blue == 1)
GoTo [loop1] (i < 10)

Color("Blue")
GoTo [loop1] (1 == 1)

loop_ends_here
```

En la línea 1 se inicializa a Wall-E en la esquina superior izquierda del canvas. En la línea 2 la brocha se pone negra. Las líneas 3 a 7 son asignaciones de variables. En la línea 9 se declara la etiqueta loop1 que representa el inicio del ciclo. El ciclo se ejecuta 10 veces y en cada iteración se dibuja un pixel a la derecha de negro y se desplaza a Wall-E una posición a la derecha (línea 10). En la línea 12 se llama a la función que pregunta si la brocha es azul. Como en las primeras 10 iteraciones del ciclo la línea será negra, el GoTo de la línea 13 se ignora y el GoTo de la línea 14 salta hacia el inicio del ciclo en la línea 9. En la décima iteración, la variable *i* tendrá valor 10 y por tanto el GoTo de la línea 14 se ignora. Luego se ejecuta la línea 16, la brocha pasa a ser azul y como el GoTo de la línea 17 siempre es verdadero, entonces se regresa a la línea 9. En esta iteración, la brocha sí será azul y por tanto se saltar a la línea 19 terminando la ejecución del programa.

ACERCA DE

Este es mi segundo Proyecto de Programación y mi primer compilador. Mi nombre es Gabriel Pérez y soy estudiante de Ciencias de la Computación de la Universidad de La Habana. Este proyecto fue creado en *Unity 6* con lenguaje C#.

El proyecto está estructurado por los siguientes scripts:

- **MainScript**: Es el que lleva el control del programa, contiene los métodos de compilar, redimensionar y genera el canvas del tamaño *large*.

- **Parser**: Contiene los métodos que reciben texto y lo separan en listas de tokens, estos son Parsing, MethodParsing, BoolParsing, PredicateParsing, y Comparation, este último actúa como un diccionario que posee los posibles tokens que puede interpretar el programa.

- **Interpreter**: Mediante el método recursivo TokenInterpreter que recibe la lista de tokens, le da sentido al código, moviéndose entre los tokens, traduciéndolos según sus parámetros o tokens próximos. Tambien contiene los métodos para resolver las operaciones booleanas y aritméticas.

- **Methods**: Es el que identifica la instrucción llamada desde *Interpreter* y la ejecuta con los parámetros que necesita. Aquí están implementados todos los métodos que aparecen en las Reglas (Instrucciones y Funciones).

- **Cells**: Script que genera el tipo de variable Cells que rellena la matriz de tamaño *large* llamada boardCells, esta variable solo contiene un parámetro, el color.

- **Menu**: Contiene el funcionamiento de la mayoría de los botones del programa.