Code to Paint

Manual de Usuario.

David Eusebio Cazares Pérez

El objetivo del documento es proveer a las personas interesadas en el uso del lenguaje las nociones básicas para poder desarrollar en el mismo. La sintaxis del lenguaje es muy parecida a c++, por lo que un usuario cotidiano del lenguaje es capaz de inferir la mayoría de las reglas.

Primeramente el lenguaje cuenta con 3 diferentes tipos de datos:

* Booleanos: True o False.
* Enteros: valores sin parte decimal.
* Dobles: valores con parte decimal.

Los primeros son nombrados como boolean, los segundos como int y los últimos como doubles. Esto es que si se quiere declarar una variable es necesario utilizar las referencias dentro de éste mismo párrafo. Es decir, una variable booleana se declara como : “boolean a”.

Otro elemento importante es que al terminar cada línea de código es necesario utilizar el punto y coma, “;”, dado que el parser lo utiliza para determinar que la instrucción ha acabado. Por lo que la llamada anterior quedaría como “boolean a;”

Un factor importante es que la comienzo de cada programa se debe escribir la palabra program seguido de un id para diferencias el programas. Entre demás reglas que se deben seguir es que el programa debe finalizar con la palabra end seguida de dos puntos. Cada programa debe tener un main ya que es la parte principal de donde el programa comienza su ejecución. El uso de llaves, {}, es necesario para las diferentes bases del programa como lo son ciclos, condicionales, funciones, etc; de no llevarlas el programa no aceptará la sintaxis.

Entre las principales líneas que se pueden escribir están:

* int a; double b; boolean c; - Es posible la declaración y asignación de un valor en la misma línea. Por lo que el programa admite sintaxis como: int a = 10; etc. Existe la posibilidad de declarar variables globales y locales.
* int a[100]; - para la declaración de variables dimensionadas. Si se le asigna un valor a una variable direccionada, en la declaración, se inicializarán todas las celas con dicho valor. En la declaración solo se pueden usar constantes enteras para definir el tamaño de la dimensión.
* A[1][2][3] – Para acceder a alguna casilla en específico de la variable. Es posible usar variables en vez de números para indexar la variable. La primera casilla siempre es considerada como a[0].
* A = 100; - Para asignar un valor a una variable previamente declarada. De asignar un valor diferentes a una casilla declarada el valor se ajustará al tipo de casilla, sin embargo no se puede realizar dicha maniobra con booleanos, solo entre enteros y dobles.
* function int fun(int b); – Para la declaración de funciones. El nombre de la función siempre debe ser precedida por la palabra function y el valor de retorno de la función. Todas las funciones deben tener un valor de retorno. Las funciones pueden o no tener parámetros, pero de llevar siempre debe especificarse el tipo de la variable.
* fun(a); - Para la llamada a la función. No se permite el uso de polimorfismo y los parámetros siempre deben ir como se declararon, no se pueden mezclar tipos. Los parámetros permitidos para pasar solo son variable lineales.
* {estatuto; return 0;}- El cuerpo de la función puede contener cualquier tipo de estatuto se desee pero al final debe llevar un y solo un estatuto de retorno. Así mismo las llamadas a funciones siempre deben agarrar el valor regresado por un variable y realizar algo con él o simplemente guardarlo en un temp.
* main {estatutos;}- El main siempre debe existir dentro del programa y debe estar rodeado de llaves.
* if (exp){estatuto;}else{estatuto;}- La utilización del else es totalmente opcional solo se debe recordar que deben estar rodeados de llaves para indicar el pedazo de código que les pertenece y las expresiones que utilizan siempre deben ser de tipo booleanos.
* while(exp){estatuto;}- Muy parecido al if y con las mismas características.
* Do {estatuto;} while(exp);- Comparte características con el while con el excepción que posee la palabra do.
* A + b; a \* b; a & b; - Todas las operaciones deben hacerse con la misma estructura solo cambiando el operador. Entre operadores usables se encuentran +,\*,/,|,&, ==, >,<. Todos son usados de la misma manera que en c. En el caso de la suma y resta un valor entero con uno doble regresará un valor entero. Las operaciones relacionales se pueden usar entre enteros y booleanos sin distinción. En éste caso el valor negativo será tomado como falso y verdadero de otra forma.
* #comment- Permite el uso de comentarios que serán ignorados para la generación del código. El comentario cuenta hasta que la línea se acabe.
* print (exp); - Imprimirá en consola la expresión dentro.
* input(a); - Permite al usuario escribir valores y guardarlos dentro de la variable designada.

El objetivo principal del lenguaje es la opción de dibujar por medio de diferente estatutos. A continuación se definen los estatutos usadas para éste fin.

* penUp(); - Levanta la pluma para permitir movimiento sin realizar trazo alguno.
* penDown();- Baja la pluma para comenzar a pintar. De aquí en adelante los movimientos generaran trazos en la pantalla.
* penSize(cI);- Para cambiar el tamaño del pincel. Solo admite valores enteros dentro.
* penColor(r,g,b); -Usada para cambiar el color del pincel. Solo admite valores enteros entre 0 y 255.
* setX(exp);- Posiciona el cursos sobre la coordenada. No genera dibujo.
* setY(exp);- Posiciona el cursor sobre la coordenada. No genera dibujo. Solo admite valores enteros.
* moveOnX(exp);- Mueve el cursor sobre el eje de las x. Solo valores enteros y no cambia la dirección del pincel. Se pueden usar valores negativos.
* moveOnY(exp);- Mueve el cursor sobre el eje de las Y. Solo valores enteros y no cambia la dirección del pincel. Se pueden usar valores negativos.
* moveForward(exp); - Se mueve cierta cantidad de unidades hacia donde el pincel este apuntando.
* rotateToRight(exp) y rotateToLeft(exp)- Ambos estatutos mueven hacia donde apunta el cursor cierta cantidad de grados. Hacia la derecha y la izquierda respectivamente. Solo admiten valores enteros.
* rectangle(base, height); - Genera un rectángulo hacia donde esté apuntando el pincel. Solo admite valores enteros.
* triangle(size); - Genera un triángulo equilátero hacia donde esté apuntando el pincel. Solo admite valores enteros.
* circle(r); -Genera un circulo de radio r. Solo admite valores enteros.

Con los estatutos antes mencionados es posible generar una gran cantidad de dibujos. Al finalizar el programa la salida visual será guardaba en un archivo con extensión ews llamado drawing.

Finalmente un ejemplo de programa realizado con la sintaxis básica.

program draw;

function int sphere(int d){

circle(d);

return 0;

}

main{

int radius = 5;

int first = 80;

int second = 60;

int third = 40;

int fourth = 20;

int trunkH = 30;

int trunkW = 20;

int c;

penUp();

moveOnY(-50);

#trunk

penDown();

rectangle(trunkW, trunkH);

penUp();

moveOnY(trunkH);

moveOnX( 0 - (first - trunkW) / 2);

penDown();

c = sphere(radius);

triangle(first);

penUp();

moveOnX(first);

penDown();

c = sphere(radius);

penUp();

moveOnX( 0 - first/2);

moveOnY(first/2);

moveOnX( 0 - second / 2);

penDown();

c = sphere(radius);

triangle(second);

penUp();

moveOnX(second);

penDown();

c = sphere(radius);

penUp();

moveOnX( 0 - second/2);

moveOnY(second/2);

moveOnX( 0 - third / 2);

penDown();

c = sphere(radius);

triangle(third);

penUp();

moveOnX(third);

penDown();

c = sphere(radius);

penUp();

moveOnX( 0 - third/2);

moveOnY(fourth/2);

moveOnX( 0 - fourth / 2);

penDown();

c = sphere(radius);

triangle(fourth);

penUp();

moveOnX(fourth);

penDown();

c = sphere(radius);

penUp();

}

end;

program fibonacci\_r;

function int fibonacci(int n){

int i = 3;

int sum = 0;

int fLast = 1;

int sLast = 1;

if (n == 1 | n == 2){

# values is 1 for fi with values 1 and 2

sum = 1;

} else {

while (i < n | i == n){

sum = fLast + sLast;

sLast = fLast;

fLast = sum;

i = i + 1;

}

}

return sum;

}

main{

print (fibonacci(7));

}

end;