Tutorial

Entry

The entry of the language DrawLab is below.

```
drawObject(Size(1000, 1200), Stroke(5))
{
    draw(Rectangle() Size(200, 200), Location(500, 500));
}
```

This program creates a Shape which is 1000 pixel width and 1200 pixel height. After then, it creates a Rectangle shape whose height and width are created according to x and y coordinate respectively.

drawObject function which is entry-point to the program, has 2 mandatory parameters, one is Size and other one is Stroke. In size, first value is the width and the second values is the height. Because of the fact that in every shape, there will be a need for width and height value, this size value is needed. Stroke values is also needed in every figure and it should be entered. Those weight, height and stroke parameters are positive integers.

Comments

A single line comment in DrawLab starts with "//". None of the characters succeeding "//" is evaluated.

```
draw(Rectangle(),Size(200,200),Location(500,500)); //This part is not evaluated
```

A multi-line comment is written between "/*" and "*/". Characters between "/*" and "*/" are not evaluated.

```
draw(Rectangle(),Size(200,200),Location(500,500));

/*

Above function draws a rectangle

These lines are comments and are not evaluated

*/
```

Types and Variables

Primitive Data Types

DrawLab will have 4 types of primitive types

Integer
String
Boolean
Float

Beside these data types, while creating a variable and assigning it to a value, user does not need to define the type of a variable explicity. Assignments will be implicit so user just

need to define a variable and assign whatever value he wants to it. In other words, it is dynamically typed.

Examples

x = “Hacı Naber?” // x is a String holding “Hacı Naber?”

// Here x is assigned to an integer values so it is now an integer 2.

x = 2;

// Here x is assigned to a float value so it is now a float 2.2.

x = 2.2;

x = .5; // x = 0.5 float

x = 465.; // x = 465.0 float

// Here x is assigned to a Color value so it is now a Color BLUE.

x = BLUE;

// Here x is assigned to a Size value so it is now a Size(3,3).

x = Size(3,3);

A variable name may start with a letter or an underscore (_), then continue with any other alphanumeric character and underscore. A variable may not contain any special character other than underscore. A variable name may be any length long.

Note that variable names are case-sensitive.

Other Data Types

DrawLab will have other 3 types of data.

Size
Location
Color

Example

// Here x is assigned to a Size value so it is now a variable whose types is Size, which has //a width 50 and height 70.

x = Size(50,70); // Only nonnegative numbers are supported to create a Size variable.

// Here x is assigned to a Location value so it is now a variable whose types is Location, //which has a x-axis 50 and y-axis 70.

x = Location(40,80); // Only nonnegative numbers are supported to create a Location // variable since origin is the left-top point of the screen.

// Here x is assigned to a Color value so it is now a variable whose types is Color, which //is a BLACK Color.

x = Color(BLACK);

// Here x is assigned to an Color value so it is now a variable whose types is Color and whose //Color is defined in terms of standard R-G-B values

x = Color(120,130,150);

Another data type that will be added is Array.

Arrays

Array data type is supported by this language. This data type is to keep the same kinds of types of data together and to keep their track. Only same data types are allowed to include in Array type. For example; a Color type variable and an integer type variable cannot be added to the same array. In addition, the length of the array has to be specified.

As the DrawLab language allows implicit declaration, user may define the variable's type as Array or he may assign it to a value or empty between the curly braces. If multiple elements are added at the time of declaration, they have to be separated by comma.

Examples:

// x implicitly assigned to an array. In this case, empty array whose length is 4.

x[4] = {};

// x implicitly assigned to an integer array. In this case, array contains the elements: 5, 4 //3 and 1. Length of the array is 4.

x[4] = {5,4,3,1};

// x implicitly assigned to an array. In this case, array contains the Color elements: //BLACK AND RED. Length of the array is 4.

x[4] = {Color. Black, Color. Red}

// x explicitly assigned to an array. In this case, the inclusion of the array is //not specified but its length is 2. Initial value of the array will be all nulls.

Array x[2];

OPERATIONS

DrawLab's Integer and Float types support some operations: Addition(+), Substraction(-), Multiplication(*), Division(\), Exponent(^).

```
// x is 6
x = 1+2+3;

// x is 1.66 as the precedence of the division is greater than the //addition
x = 1+2/3;

//x is 3.2 as the addition of float and integer results in float
x = 1.2 + 2;

//x is 125 regular exponent operation
x = 5^3;
```

Addition, subtraction, multiplication and division operations are left-associative. Power operation (^) is right-associative:

```
num1 = 8 / 4 / 2;    // 1 (not 4, this statement is equal to (8/4)/2, 8/(4/2) would
                          // make a different result
num2 = 2 ^ 3 ^ 2;   // 512 (this statement is equal to 2^(3^2), not (2^3)^2 )
```

DrawLab's Boolean types support logical operations: And(&), Or(|) and Not(!)

```
// x is false
x = false;
// x is false
x = !(true)
```

//Paranthesis has the greatest precedence so initially x|y is found. The result of x|y is true as it is an or operation.Then !(true) is calculated as !'s precedence is greater than &. !(true) is false. At last, false&true = false.

```
x = true;
```

```
y = false;

z = true;

!(x|y)&y


//x is 125 regular exponent operation

x = 5^3
```

Moreover, strings are concatenated using '+' operator. Other types are automatically converted to string when they are concatenated with a string.

```
firstName = "Bugra";

lastName = "Gedik";

courseCode = 315;

fullName = firstName + " " + lastName + " " + courseCode;    //fullName = "Bugra Gedik 315"
```

SHAPES

DrawLab includes several built-in shapes in itself, these are Line, Rectangle and Oval. They can be drawn with their several properties such as different size, different location, and their color and so on.

```
a=Rectangle();

draw(a, Size(40,40),Location(33,33));
```

LINE

Line is the one of the 3 built-in shapes that DrawLab has. When you are drawing line in this language you can choose multiple of choices that you have to give whatever type you want to your line. A Line shape can be instantiated like the following:

```
Line(TRUE);
```

Above piece of code creates a line which lies along upper-left to bottom-right diagonal of its bounding box when it is drawn. If the argument is specified as FALSE, the line lies along bottom-left to upper-right diagonal of its bounding box.

```
a = Line(true);

draw(a,Size(100,30),Location(0,0));

b = Line(false);

draw(b,Size(100,30),Location(0,0));
```

One argument for Line is mandatory. A Line cannot be instantiated without any arguments.

A line which is parallel to x axis can be drawn by setting height of bounding box as 0. Likewise, a line which is parallel to y-axis can be drawn by setting width of bounding box as 0.

RECTANGLE

Rectangle is also one of the 3 built-in shapes that DrawLab language has. When you are drawing rectangle you can do it with or without rounded corners. If you want it as rounded you have to specify it when you declare otherwise just in default it will come as normal rectangle. Initiation of rectangle can be done as follows:

```
//default rectangle with no rounded corners

Rectangle();

//rectangle with 23 degree of roundness

Rectangle(23);
```

Indeed, you can give location size and whatever you want to it by draw function as follows:

```
draw(Rectangle(22),size(30,55),location(33,68),fillColors(Color(BLACK)));
```

Where we will have rectangle with 22 degree roundness. Where it starts from 33 of x and 68 of y and size of 30 and 55. Lastly, it is filled with Black color

OVAL

Oval is the last of the built-in shape of DrawLab programming language. It is only shape that cannot be initialized with any parameter such as *Oval()* is the only way to initialize. Also it can be given size, color and so on. For example,

```
draw(Oval(),size(30,33),location(23,28));
```

Where Oval drawn with size of 30 and 33, also, location of 23 and 28.

CREATING NEW SHAPE

To create new shape that DrawLab has not got as built-in shape, one can use built in ones and create any new shape that she wants. Also, she can use it later when defining new shapes such as if she instantiated square, she can use it for creating 3x3 square. To define new shape one must write following

newObject *name*(argument1, ….)

You can put as many argument as you want which are mandatory or optional. However, you can also place this place as blank also. Instance of creating new shape stated below.

```
newObject Trapezium {
a1=Line(false);
a2=Line(true);
a3=Line(true);
a4=Line(true);

draw(a1,Size(50,50),Location(0,0));
```

```
draw(a2,Size(80,0),Location(50,50));
draw(a3,Size(50,50),Location(130,50));
draw(a4,Size(180,0),Location(0,0));
}
```

Where with help of built in shape line we draw trapezoid which is one of common figure in geometry.

DRAW FUNCTION

Draw function helps us to draw built in shapes of the system easily. It is written as stated below:

draw(argument1,argument2,….)

It has total of 6 arguments, however, 3 of them are mandatory to write, other 3 arguments are optional. The 3 mandatory arguments are type of shape, type and size of it. For instance:

draw(Rectangle(),Size(100,20),Location(0,0))

Additionally, for drawing lines size means different compared to oval and rectangle. Size means end of the line where location means start of the line.

Furthermore, we can give some optional things to shapes, such as, determining stroke width, fill it with color, and determine the color. Instance of it mentioned below:

draw(Rectangle(),Size(100,20),Location(0,0), fillColor(Color(RED)),Stroke(3),fillState(true))

One can make color with the determined ones such as Color(RED), Color(BLACK) and so on. Also, it can be given as Red Green Black (RGB) values such as 100,100,100 which means dark grey in RGB values. Additionally, stroke width can be chosen from 1 to 5 category. Lastly, fill state determines whether to fill inside of figure with this color or not.

CONDITIONAL STATEMENTS

Conditional statements are one of popular concept in programming languages. If first condition is satisfied the process is done and do not go other conditions, if not go to next check and do the same as you done for previous one. It helps us to execute something with checks such as follows:

```
a=1;

b=2;

if(a==0){

    c=a-b;

}

else if(b==0){

    c=b-a;

}

else{
```

```
    c=a+b;
}
```

LOOPS

Loop concept do the same job as its name it iterates the same lines as much as you want. It is also one of the core concept of programming and used widely.

WHILE

While is one of the basic type of loop. It iterates until the condition written is satisfied, if it is not it iterates infinitely. Example of while loop written below:

```
i=4;
a=0;
while(i<500){
    a=a-1;
    i=i+1;
}
```

FOR

For loop is relatively advanced when it is compared to while loop. Such that when you are writing for loop you have 3 parts do some value operation with existing or new value, writing your condition statement and incrementing. Instance of for loop stated below (it is replication of example above):

```
a=0;
for(i=4;i<500;i=i+1){
    a=a-1;
}
```

DEFINING FUNCTIONS

To do series of operations in DrawLab you can make functions which can do several operations under one of it. To instantiate function you have to function at the top and follow it with name you gave to it and in parenthesis you have to write the variables you want to make proper operations inside of function. However, if you want you can make this part empty. Also, every function must have return it can be with value or without value, however, we do not need to give type of return language can automatically find its type. Instance of function is written below:

```
function findMax(arr,length){
    if(arr==NULL)
        return;
    max=arr[0];
```

```
        for(i=0;i<length;i=i+1){
                if(max<arr[i]){
                        max=arr[i];
                }
        }
        return max;
}
```