CS315 – Programming Languages Project

“Paint++”

Tutorial

Course Instructor: Buğra Gedik

Group Number: 2-9

Group Members: Erin Avllazagaj

Gülsüm Güdükbay

Melis Kızıldemir

Table of Contents

1. Expressions and Statements

Paint++ ignores whitespaces. So any new line characters are ignored after or before any elements such as curly brackets. This can be seen below:

```
if ( 1 == 1){
        a = b
}
```

```
if ( 1 == 1)
{
        a = b
}
```

```
if ( 1 == 1)
{ a = b}
```

The above code blocks are all interpreted the same way, that is, they all perform and act the same.

2. Variable Allocation

2.1 Primitive Types

Type	Syntax	Description
Integer	var iintgr	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Float	var flot	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Boolean	var flag	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
String	var str	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.

2.2 Complex Types

Type	Syntax	Description
Size	var sze	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Location	var loc	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Bounding Rectangle	var brect	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Shape	var shpe	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.
Color	var colr	Since this language is dynamically typed, all of the variables are allocated with the “var” identifier.

3. Variable Instantiation

3.1 Primitive Types

Type	Syntax	Description
Integer	`var iintgr = 2`	Since the variable types are determined at runtime, the instantiation determines a "var"s type according to the type of right side of assignment.
Float	`var flot = 3.4`	The right hand side should have a decimal point.
Boolean	`var flag = TRUE`	The right hand side can also be 1, 0, false or logical expressions too.
String	`var str = "Hello!"`	The right hand side should be in quotation marks.

3.2 Complex Types

Type	Syntax	Description
Size	`var sze = Size(wid, len)`	Can access the length and width. Is used in BoundingRectangle instantiation.
Location	`var loc = Location(x,y)`	Can access the x–coordinate and y–coordinate. Is used in BoundingRectangle instantiation.
Bounding Rectangle	`var br = BoundingRectangle( loc, sze)`	Uses the Size and Location instances to determine the size and location of the drawn object.
Color	`var colr = Color.rgb( 0,0,255)`	Color.rgb accepts 3 integer type variables (constant or not) and returns a Color object.
Line	`var ln = Line(NW, 3, 1)`	Line accepts an enumeration like NW, N, S, E, W, NE, SW or SE in the first parameter for the direction of the line. For second parameter, it accepts one of 0, 1 or 2 which indicate the arrow is in the arrow is in the starting end, if it's given 1, the arrow is in the ending end and if it is given 2, the arrow is in the both ends. The third parameter is the ratio of the arrow size to stroke width. It's an int.
Oval	`var ovl =  Oval()`	There are no parameters needed in this variable type.
Rectangle	`var rect = Rect(1)`	The only parameter required for this type is the rounded corner specification. If the programmer gives 1, the rectangle is drawn with rounded corners. If its zero, its drawn regularly.

Composite	`var br = BoundingRectangle( loc, size)`
	`var rect = Rect(0)`
	`var comp`
	`var arr`
	`var i`
	
	`while(i < 12)`
	`{`
	`    arr[i] = Rect(1)`
	`    i = i + 1`
	`    arr[z]= Location(200,200+i)`
	`    i = i + 1`
	`    arr[z] = Size(50, 50)`
	`    i = i + 1`
	
	`    arr[i] = tbf`
	`    i = i + 1`
	`    arr[i] = True`
	`    i = i + 1`
	`    arr[9] = Color(0, 0, 255)`
	`}`
	`comp = Composite(arr)`
	`Draw(comp, boundingRect, strokeWidth)`
	`i = i + 1`
		To instantiate a composite shape, one needs to create a one–dimensional array and sequentially put each shape variable and its properties by specified order in the syntax and then, the Composite function takes this array as a parameter and returns a Composite shape. So each property should be given in the same order for the identification of each shape.

4. Operators

4.1 Logical Operators

Operator Name	**Syntax**	**Description**
AND	a && b	Is a binary logical operator for AND operation.
OR	a || b	Is a binary logical operator for OR operation.
NOT	!a	Is a unary right associative operator for NOT operator.
Equals	a == b	Is a binary logical operator to determine whether "a" is equal to "b".
Not Equal	a != b	Is a binary logical operator to determine whether "a" is not equal to "b".
Less Than	a < b	Is a binary logical operator to determine whether "a" is less than "b".
Less Than or Equal To	a <= b	Is a binary logical operator to determine whether "a" is less than or equal to "b".
Greater Than	a > b	Is a binary logical operator to determine whether "a" is greater than "b".
Greater Than or Equal To	a >= b	Is a binary logical operator to determine whether "a" is greater than or equal to "b".

Precedence: && > || > ! > ((==) = (!=) = (<) = (<=) = (>) = (>=))

4.2 Arithmetic Operators

Operator Name	**Syntax**	**Description**

Addition	var innt
	var floaat
	var strr
	innt = 2 + 3
	floaat = 4.5 + 1.2
	strr = "Hello" + "World!"
		The int, float, boolean and string types support addition. Besides number addition, strings are added together for concatenation purposes.

Subtraction	var innt
	var floaat
	var booll
	innt = 2 - 3
	floaat = 4.5 - 1.2
	booll = TRUE - FALSE
		The int, float and boolean types support subtraction

Division	var innt = 3/4 var floaat = 3.4/2 floaat = 4.5 / 1.2	The int and float types support division.
Multiplication	var innt = 2*3 var floaat = 1.2 *3.2	The int and float types support multiplication.

5. Drawing Methods

Shape Type To Be Drawn	**Syntax** (strokeWidth is int, fillState is bool, fillColor is Color)	**Description**
Rectangle	Draw (shape, boundingRectangle, strokeWidth, fillState, fillColor)	For all the Rectangle type, if the fillState is given FALSE (0), the fillColor is ignored. Otherwise, for the Rectangle, all of the parameters are mandatory.
Oval	Draw (shape, boundingRectangle, strokeWidth, fillState, fillColor)	For all the Oval type, if the fillState is given FALSE (0), the fillColor is ignored. Otherwise, for the Oval, all of the parameters are mandatory.
Line	Draw (shape, boundingRectangle, strokeWidth, fillColor)	All of the parameters are mandatory.
String	Draw(str, boundingRectangle, fillColor)	All of the parameters are mandatory.
Composite	Draw (shape, boundingRectangle, strokeWidth, fillState, fillColor)	For the Composite type, strokeWidth, fillState and fillColor are optional parameters, because, if the programmer doesn't specify one or more of them, the interpreter will understand that it should take the colors or fill state or stroke width of the individual unit shapes that make up the composite shape.

6. Conversions

Operator Name	Syntax	Description

To int	var innt
	var floaat
	var strr
	var boool
	
	floaat = 3.4
	innt = int(float) //truncating after decimal point, 3
	
	strr = "54"
	innt = int(strr) //54
	
	boool = TRUE
	innt = int(boool) // 1
		The string, float and bool types support conversion to int. When a float is converted to int, the number is truncated after the decimal point. For bool types, the variable is converted to 1 for TRUE and 0 for FALSE. For string, the variable in the quotations should be numbers, or error message will be raised.

To float	var innt
	var floaat
	var strr
	var boool
	
	innt = 3
	floaat = float(innt) //extending after decimal point, 3.0
	
	strr = "5.4"
	floaat = float (strr) //5.4
	
	boool = TRUE
	floaat = float (boool) // 1.0
		The string, float and bool types support conversion to float. When a int is converted to float, the number is 0 extended after the decimal point. For bool types, the variable is converted to 1.0 for TRUE and 0.0 for FALSE. For string, the variable in the quotations should be floats or ints, or error message will be raised.

To bool	var innt
	var boool
	innt = 0
	boool = bool(innt) //FALSE
	
	var flot
	var boool
	flot = 2.8
	boool = bool(flot) //TRUE
	
	var str
	var boool
	str = "TRUE"
	boool = bool(str) //TRUE
		The integers, floats and strings also support conversion to boolean. However, for string, the programmer should give "TRUE" or "FALSE" as parameter.

To string	var innt
	innt = 1
	var str
	str = str(innt)
	
	var flot
	flot = 1.4
	var str
	str = str(flot)
	
	var bool
	bool = 0
	var str
	str = str(bool) //”FALSE”
		The integers, booleans and floats can be converted to strings.

7. Conditionals

This language supports conditionals, as if/ if and else statements, that is, matched and unmatched if statements are supported. The syntax is as follows:

```
if( <logical_expression>) { //do stuff here }

if( <logical_expression>) {
        //do stuff here
}

if( <logical_expression>)
{
        //do stuff here
}

if( <logical_expression>){
        //do stuff here
}
else {
        //do stuff here
}
```

Please note that the position of the curly brackets doesn’t change the interpretation, however the curly brackets and the brackets are mandatory.

8. Loops

This language only supports while–loops. The syntax is as follows:

```
while (<logical_expression>)
{
        //do stuff here
}
```

Note that the curly brackets around the body of the loop and the normal brackets in the while statement around the logical expression are mandatory, even if the programmer should write one line of body!

9. Comments

This language only supports one – line comments. So, the programmer should follow the following syntax while writing a comment.

```
//INSERT YOUR COMMENT HERE
```