Melhor Linguagem para Iniciantes - Parte III (Final)

Mário Leite

...

Na **Parte II** desta postagem foram mostradas três opções de ações do menu principal do IDE do Small Basic: *Importar/Publicar/Graduar*; esta última, na realidade não tem muito a ver com "graduação"; é para fazer a conversão de Small Basic para Visual Basic. Nesta terceira (e última) parte vamos trabalhar um pouco mais com o aspecto de linguagem orientada a objetos desta linguagem, sem no entanto, aprofundarmos neste assunto. Depois, também de maneira bem introdutória, trataremos dos recursos gráficos da linguagem e mostrando o seu lado lúdico, já que esta linguagem é muito indicada para a introdução de crianças (dos 11 aos 16 anos) no mundo da programação pois, a possibilidade de gerar figuras é um atrativo muito forte para este tipo de usuário.

Quando queremos ler um valor digitado as instruções podem ser as seguintes:

```
TextWindow.Write("Digite o valor do raio: ") 'mensagem explicativa
raio = TextWindow.Read() 'lê o valor da variável digitada
```

A primeira linha de código é uma mensagem esclarecedora para o usuário saber qual dado deve ser digitado; a segunda lê (armazena na memória) o valor digitado: no caso, o raio de um círculo. E para calcular e mostrar a área do círculo cujo valor do raio foi dado na variável ***raio***, podemos utilizar as duas linhas de código mostradas abaixo.

```
area = 3.14159265358979323846264338279502884197169399375106*(raio*raio)
TextWindow.WriteLine("Area do círculo: " + area)
```

A **figura 1** mostra a saída do programa com estas quatro linhas, onde o objetivo é mostrar a área de um círculo de raio lido pelo teclado.

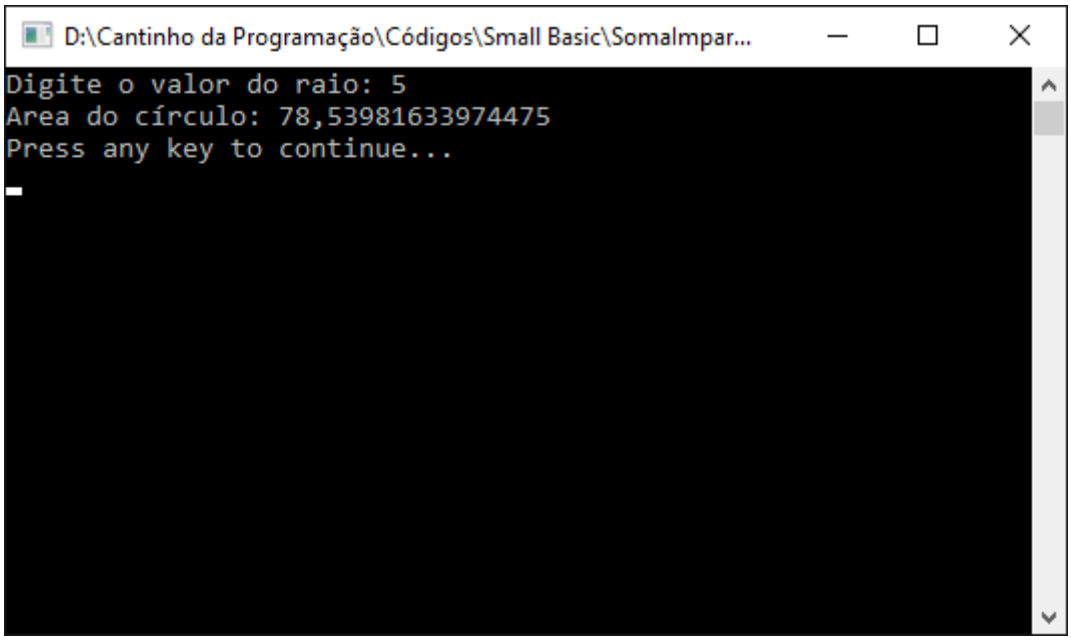


Figura 1 - Primeira saída do programa

Observe que, mesmo usando PI com cinquenta decimais a área do círculo saiu com quatorze; mas, caso o usuário não precisar de tantas decimas o Small Basic oferece recursos para arredondar o resultado. Por exemplo, se for necessário calcular a área com apenas quatro decimais, ficaria assim: inserindo a quarta linha para fazer o arredondamento necessário.

```
TextWindow.Write("Digite o valor do raio: ")  'mensagem explicativa
raio = TextWindow.Read()  'lê o valor da variável digitada
area = 3.14159265358979323846264338279502884197169399375106*(raio*raio)
area = Math.Round( area * Math.Power(10, 4) ) / Math.Power(10, 4)
TextWindow.WriteLine("Area do círculo: " + area)
```

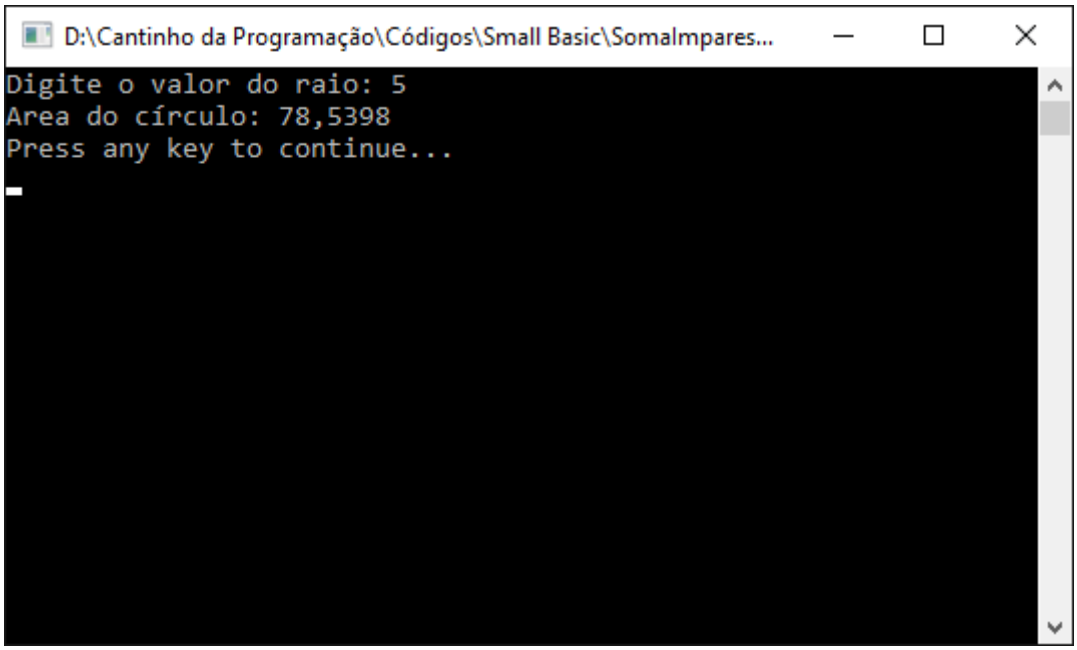
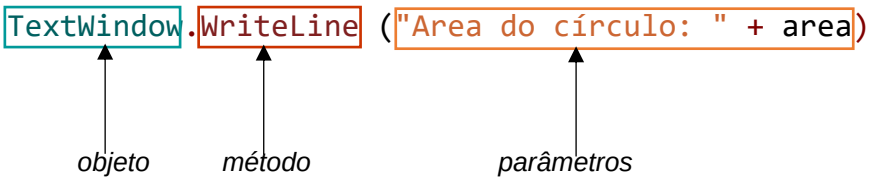


Figura 2 - Segunda saída do programa

Nas linhas de código acima podemos destacar as seguinte palavras-chave: TextWindow, Math, Write, WriteLine, Read, e Roud. Observe as diferentes cores para estas palavras-chave: na cor *verde azulada* temos os *objetos* e na cor *marrom* os *métodos*. Nas linguagens não OO os objetos seriam os *tipos de dados* e os métodos as *rotinas*. Assim, o raio no programa acima seria do tipo inteiro (***int*** em C) e ***Integer*** em Pascal. Então Write, WriteLine, Read e Roud são *métodos* (operações) do objeto TextWindow; o que vem dentro dos parênteses são os parâmetros exigidos pelo método (para serem processados). O esquema abaixo explica...



Ao digitar uma palavra-chave (que represente um objeto) e em seguida um ponto (**.**) o Small Basic mostra diversos termos que representam os *membros* da classe desse objeto. Portanto, ao digitar a palavra Math seguida de um ponto (**.**) são oferecidos vários membros (*métodos* e *propriedades*) do objeto Math. A **figura 3** mostra alguns *métodos* da classe deste objeto; Roud é apenas um deles, e que faz o arredondamento de um número do tipo ***real*** (com decimais).

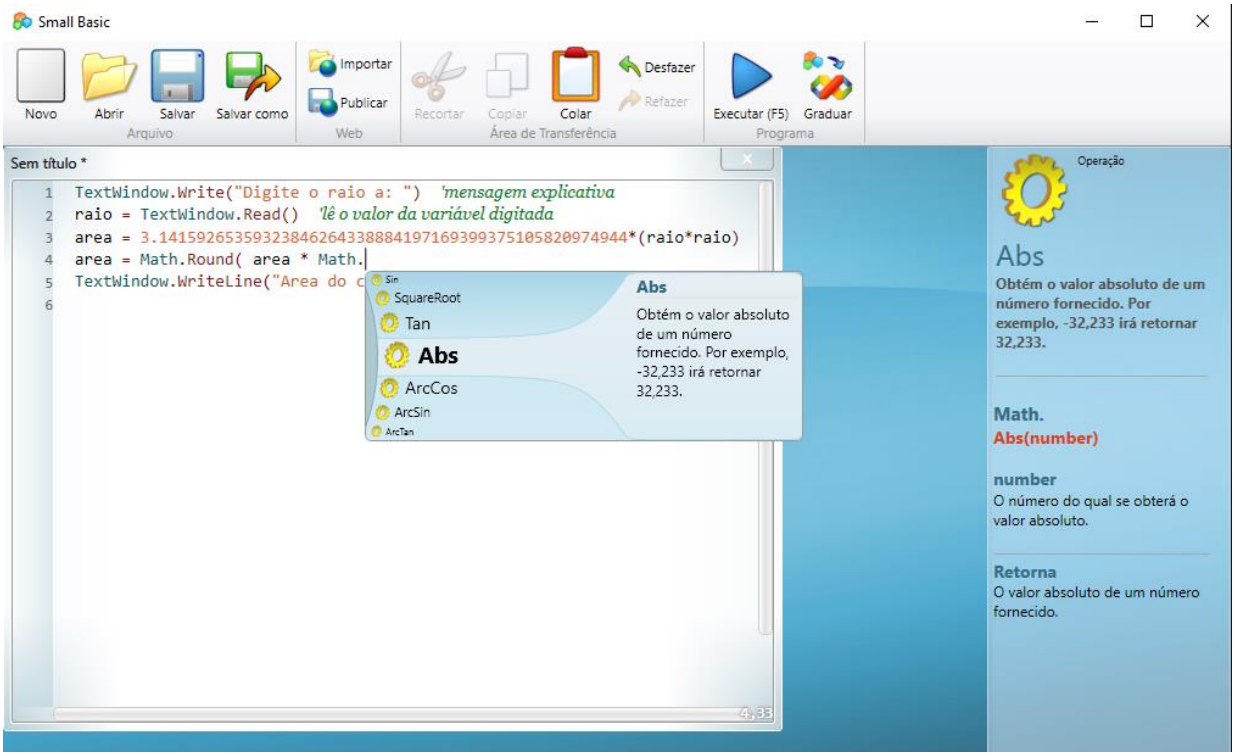


Figura 3 - Mostrando os membro do objeto Math

O objeto TextWindow permite trabalhar com textos na janela “Window”; mas, caso o usuário queira trabalhar com elementos gráficos é necessário crar um ambiente para isto: janela gráfica. O objeto base para ctiar e manipular elementos gráficos no Smal Basic é **GraphicsWindow**, traduzido literalmente como “janelas gráfica”. Para criar uma janela gráfica funcional as instruções básicas são as seguintes:

```
GraphicsWindow.BackgroundColor = "cor de fundo da janela"  'opcional
GraphicsWindow.Title  = "Título da Janela"
GraphicsWindow.Width  = #largura da janela# em pixels
GraphicsWindow.Height = #altura da janela## em pixels
GraphicsWindow.Show()  'exibe a janela
```

Nesta caso temos os seguintes elementos de OOP:

GraphicsWindow ==> *objeto*
BackgroundColor, Title, Width, Height ==> *propriedades*
Show() ==> *método*

Vamos considera as seguintes linhas de código

```
GraphicsWindow.BackgroundColor = "Green"
GraphicsWindow.Title = "Minha Janela Gráfica"
GraphicsWindow.Width = 380
GraphicsWindow.Height = 250
GraphicsWindow.Show()
```

Acima são mostradas as propriedaes **BackgroundColor**, **Title**, **Width** e **Heighr**, além do método **Show()**; mas existem muitos outros membros do objeto GraphicsWindow; entre eles podemos destacar os segintes
:

PenWidth ==> propriedade que define a espessura da caneta que traça linhas.
PenColor ==> propriedade que define a cor da caneta que traça linhas.
DrawLine() ==> método que traça uma linha.

A **figura 4** mostra a saída da execução das cinco linhas de instruções mostradas anteriormente.

Figura 4 - Um exemplo de janela gráfica

O código abaixo é outro exemplo de código de programa para traçar um círculo no centro da janela gráfica da **figura 5**, com circunferência na cor amarela de raio 80..

```
GraphicsWindow.BackgroundColor = "Green"
GraphicsWindow.Title = "Minha Janela Gráfica"
GraphicsWindow.PenWidth = 3 'define a espessura da caneta
GraphicsWindow.PenColor = "Yellow" 'define a cor da circunferência
GraphicsWindow.Width = 600  'coordenada x do centro do círculo
GraphicsWindow.Height = 400 'coordenada y do centro do círculo
raio = 80
GraphicsWindow.DrawEllipse(300 - raio, 200 - raio, raio * 2, raio * 2)
```

Figura 5 - Círculo traçado no centro da janela

Basedo na **figura 5** podemos traçar círculos concêntricos para criar um efeito visual interessante; basta colocar a instrução que traça o círculo dentro de um *loop*, variando as coordenadas do centros em função da variável de controle do *loop*. Observe como isto pode ser feito no código abaixo.

```
GraphicsWindow.BackgroundColor = "Green"
GraphicsWindow.Title = "Minha Janela Gráfica"
GraphicsWindow.PenWidth = 3  ' espessura da caneta
GraphicsWindow.PenColor = "Yellow"
GraphicsWindow.Width = 600
GraphicsWindow.Height = 400
For raio = 1 To 100 Step 12
  GraphicsWindow.DrawEllipse(300 - raio, 200 - raio, raio * 2, raio * 2)
EndFor
```

A saída deste programa é mostrada na **figura 6**: doze círculos concêntricos com centro na posção (300,200) e na cor amarela.

Figura 6 - Círculoa concêntricos no centro da janela

Além de *propriedades* e *métodos* o Small Basic também oferece *eventos* (tal como nas linguagens Orientadas a Eventos, como C#, Visal Basic, Java). Esses eventos são ligados a ações disparadas sobre a janela gráfica, o que aumenta a interatividade **usuário-programa.** Entre esses eventos podemos destacar alguns:

Tick:	controle do crinômetro.
KeyDown	quando alguma tecla é pressionada.
KeyUP	quando uma tecla é liberada.
MouseDown	quando o botão do *mouse* é pressionado.
MouseUp	quando o botão do *mouse* é liberado.
MouseMove	quando o botão do *mouse* é pressionado e arrastado sobre a janela.
TextInput	quando algum texto é digitado na janela gráfica.

A janela gráfica é um recurso fundamentel no aprendizado da linguagem Small Basic; na verdade é o atrativo principal para os iniciantes, principalmente para crianças que querem entrar no mundo da codificação com uma linguagem fácil de aprender, mesmo com recursos limitados. Neste contexto, um objeto muito explorado é a famosa tartaruguinha, definida no objeto **Turtle**, que é uma alternativa á linguagem Logo. Os principais membros deste objeto são os seguintes:

Speed	propriedade que define a velocidade com que o objeto se move.
Angle	propriedade que dá o ângulo atual da tartaruga.
X	propriedade que define a coordenada horizontal (em pixels) da posição da tartaruga.
Y	propriedade que define a coordenada vertical (em pixels) da posição da tartaruga.
Show()	método que mostra e habilita a tartaruga.
Hide()	método que oculta e desabilita a tartaruga.
PenDown()	método que abaixa a caneta para que a tartaruga possa desenhar ao se mover.
PenUp()	método que levanta a caneta para que a tartaruga não desenhe ao se mover.
Move(dist)	método que move a tartaruga da distância (*dist*) especificada. Se a caneta estiver abaixada vai sendo desenhada uma linha.
MoveTo(x,y)	método que move a tartaruga para as coordenadas (*x*,*y*) especificadas. Se a caneta estiver abaixada vai sendo desenhada uma linha.
Turn(ang)	método que gira a tartarua de um ângulo especificado.
TurnRight()	método que gria a tartaruga de 90º para a direita.
TurnLeft()	método que gria a tartaruga de 90º para a esquerda.

As linhas de código abaixo, quando executadas, mostram uma janela de fundo branco e com a tartaruga no seu centro, como pode ser visto na **figura 7a**.

```
'Configuração da janela
GraphicsWindow.BackgroundColor = "White"
GraphicsWindow.Title = "Minha Janela Gráfica"
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300
GraphicsWindow.Show()

' Configuração da tartaruga
Turtle.X=190
Turtle.y=125
Turtle.Show()
```

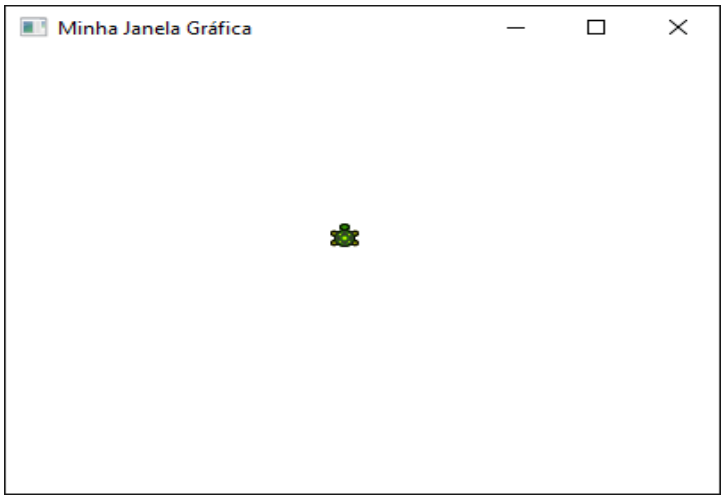


Figura 7a - Mostrando a tartarura no centro da janela gráfica

Depois de mostrar a tartaruga, ao executar a linha de instrução ***`Turtle.Move(80)`*** ela se desloca 80 pixels para cima traçando uma linha (com a *pen* abaixada). Veja na **figura 7b**.

Figura 7b - A tartarura se movento 80 pixels para cima

Finalmente, é importante esclarecer que nestas três partes da postagem "*Melhor Linguagem para Iniciants*" o objetivo não foi o de apresentar um tutorial sobre Small basic mas, sim de mostrar que, embora tenha muitas limitações, pode-se deduzir que é uma excelente opção para os iniciantes em programação que querem aprender uma linguagem real que os introduza no ambeinte de codificação, sem precisar enfrentar as complexidades de linguagens mais avançadas, e mesmo assim aprender a programar de maneira fácil e rápida.