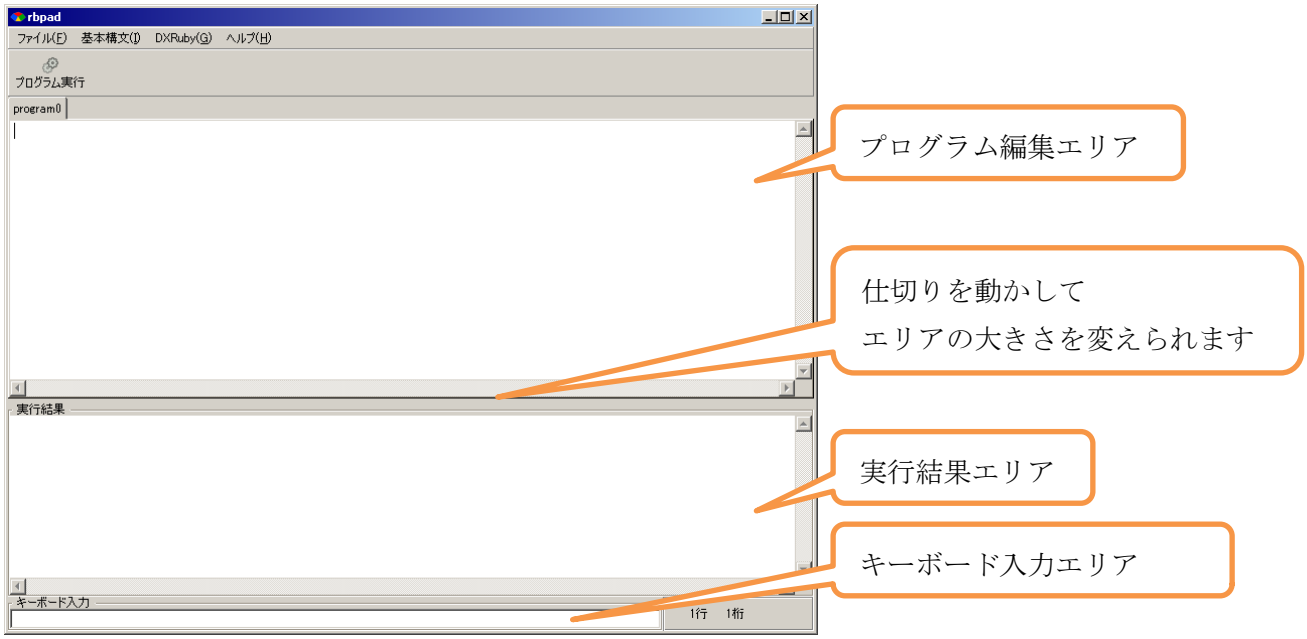
『rbpad』を使ってみよう！

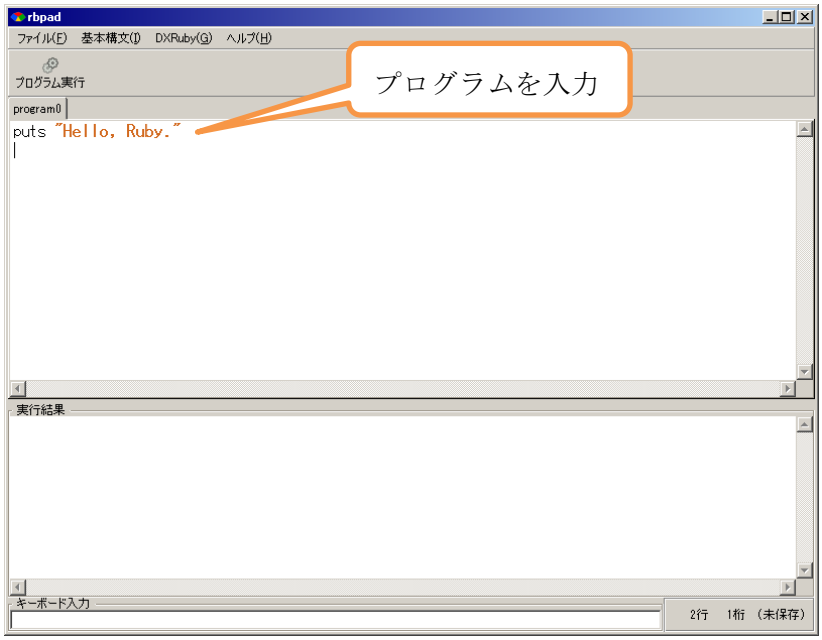
1．まずは試してみよう

(1)rbpad を起動して、プログラム編集エリアに Ruby のプログラムを書いてください。



[記述例]

```
puts "Hello, Ruby."
```

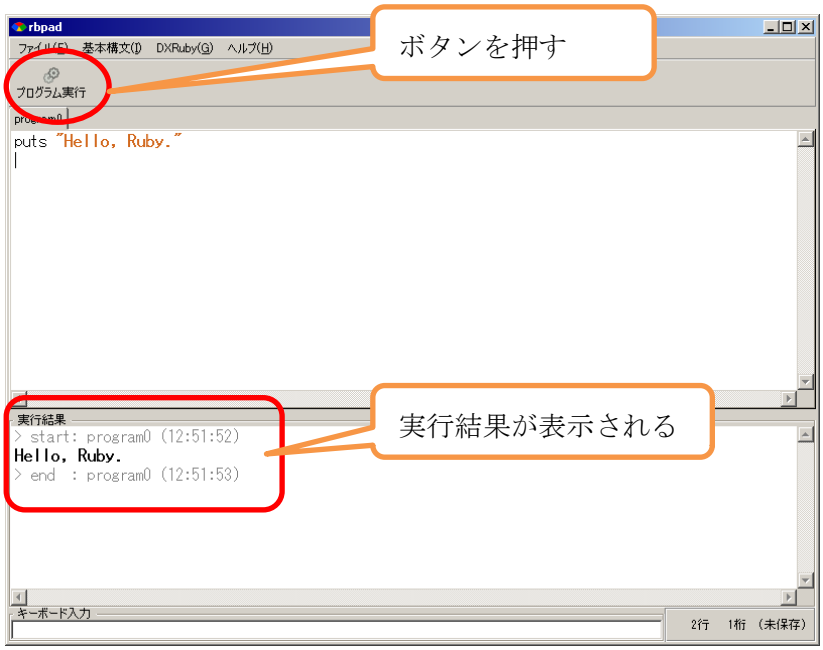


(2)ツールバーの [プログラム実行]ボタンをマウスでクリックしてください。
(または、メニュー[ファイル]－[プログラムを実行する]を選ぶか、
キーボードで [Ctrl]＋[R]キーを押してください)

(3)実行結果エリアに、プログラムの実行結果が表示されます。
(プログラムの実行開始と実行終了の時刻も、その前後の行に自動的に表示されます)

[表示例]

```
> start: program0 (12:51:52)
Hello, Ruby.
> end   : program0 (12:51:53)
```



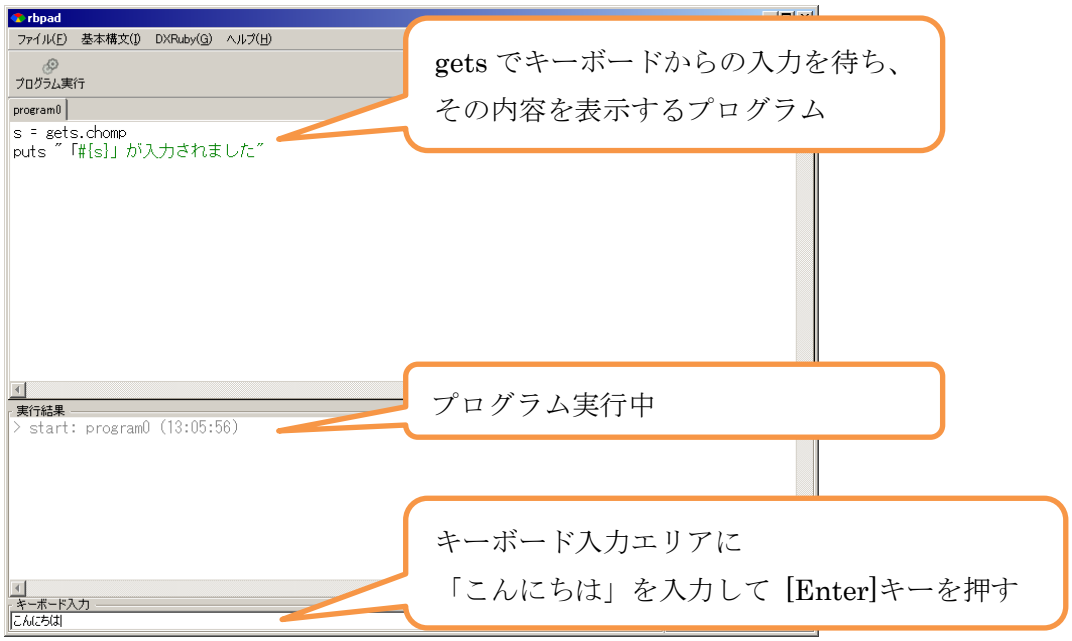
(4)基本的な使い方はこれだけです。
いろいろなプログラムを書いて、どんどん実行してみましょう。

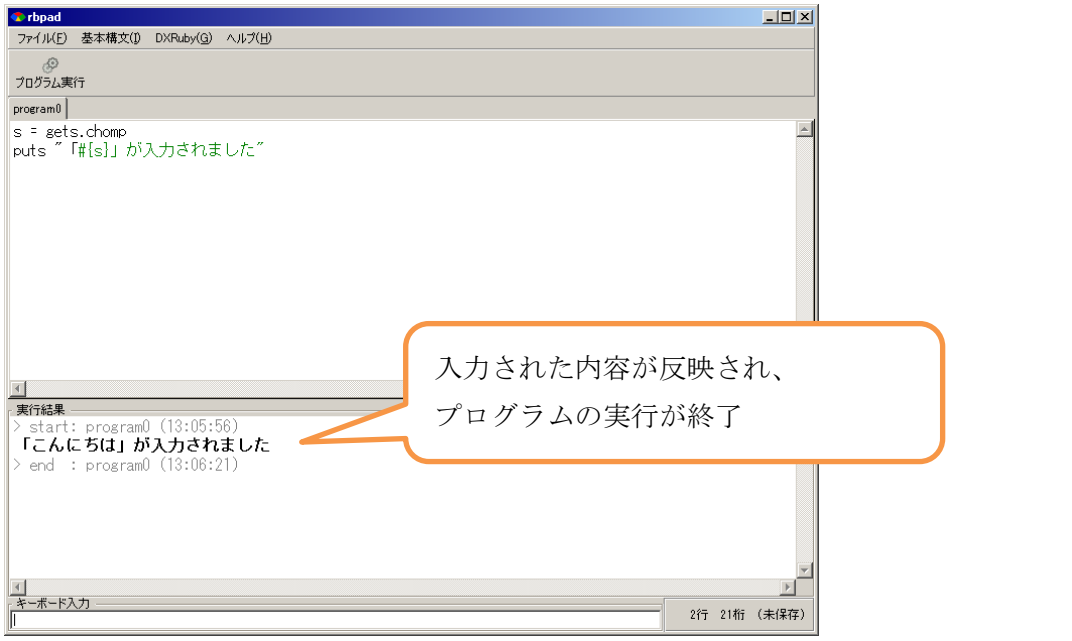
（補足）

gets メソッドなどでキーボードからの入力を受け付けるプログラムを実行した場合、
キーボード入力エリアに文字を入力して [Enter]キーを押すことによって
実行中のプログラムに入力情報を送ることができます。

[記述例]

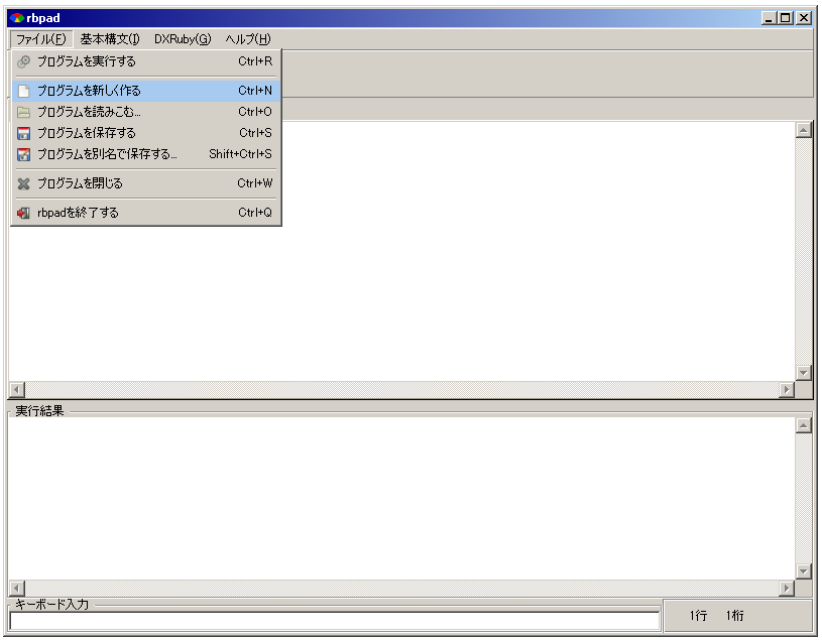
```
s = gets.chomp
puts "「#{s}」が入力されました。"
```

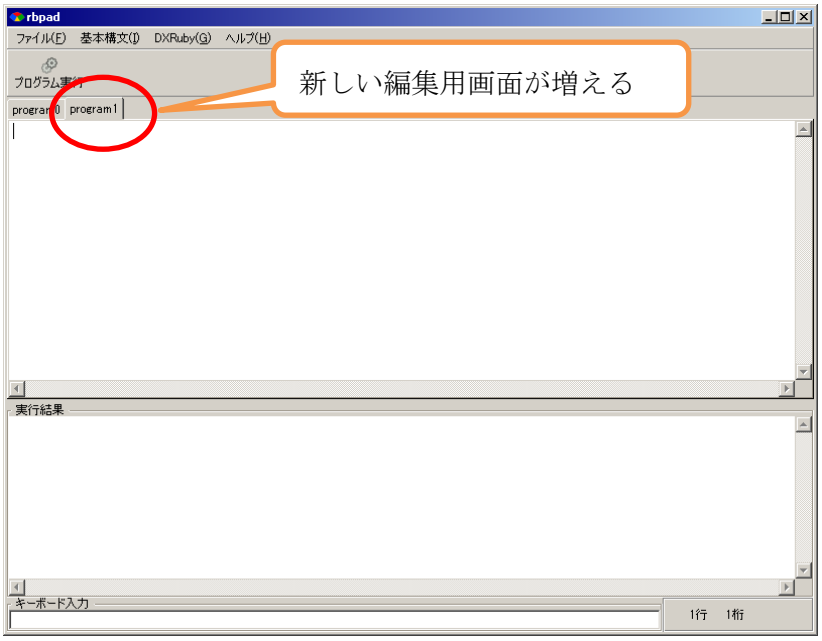




２．プログラムをどんどん書いてみよう

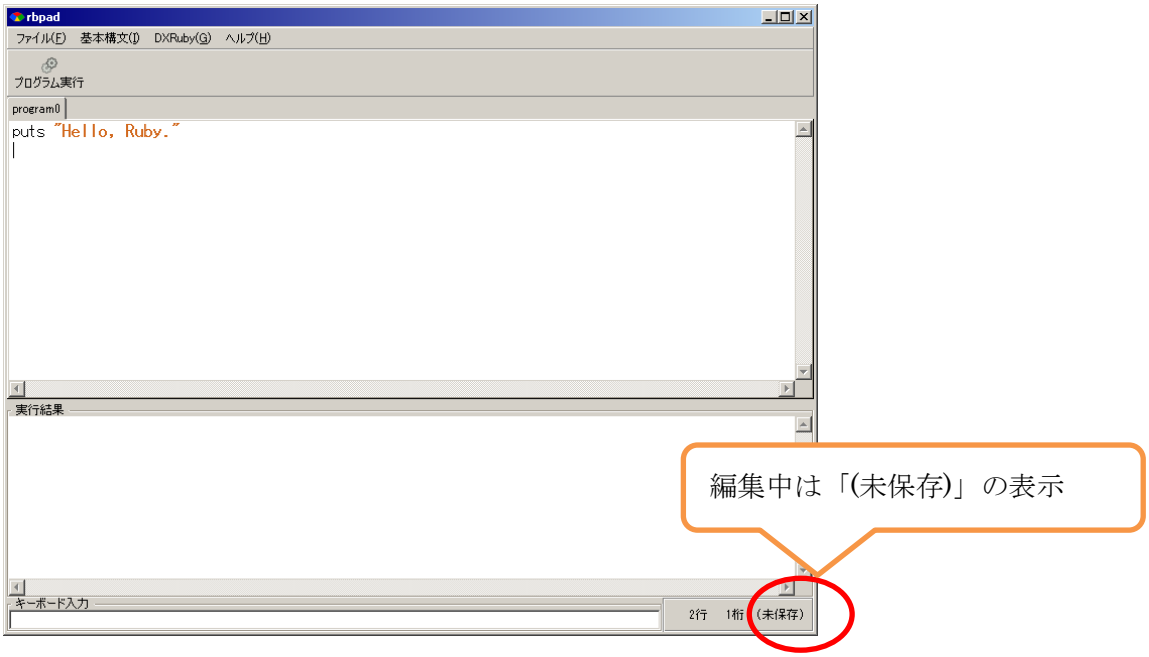
(1)メニューの [ファイル]－[プログラムを新しく作る]を選ぶと、プログラム編集エリアに
タブが１つ増えて、新しいプログラムを書くことができるようになります。
(キーボードで [Ctrl]＋[N]キーを押しても同じ動作になります)



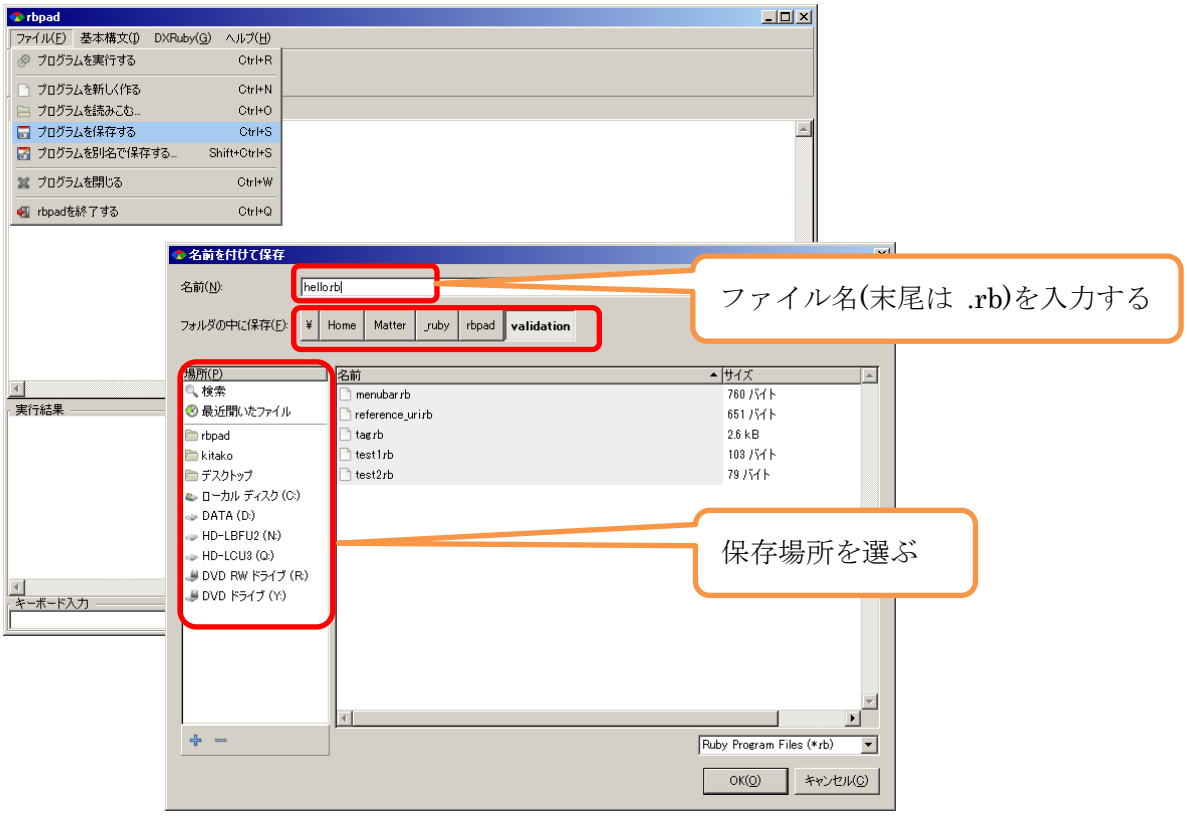


「こう書いたらプログラムはどう動くのかな？」と思ったら、タブをどんどん増やしながら
いろいろなバリエーションのプログラムを実際に書いてみて、実行してその動きを確認して
みてください。

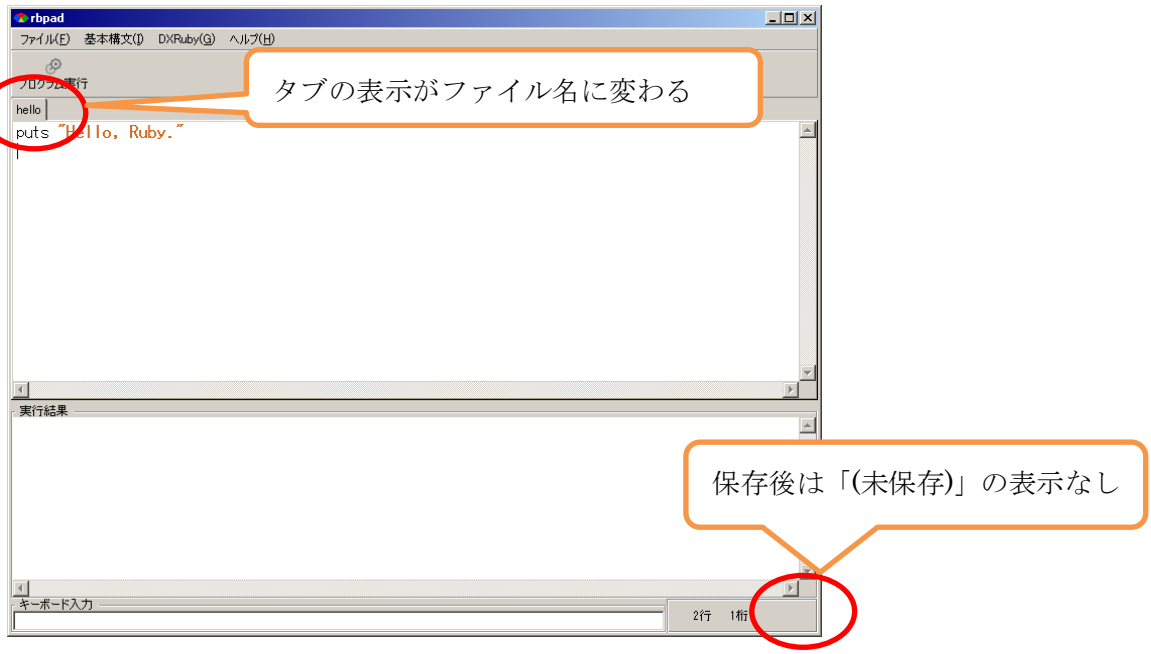
(2)メニューの [ファイル]−[プログラムを保存する]を選ぶと、入力したプログラムに
名前を付けてファイルに保存することができます。
(キーボードで [Ctrl]＋[S]キーを押しても同じ動作になります)



[プログラムを保存する]を選ぶと、いちばん最初の保存のときには保存場所とファイル名を
指定するダイアログが開かれます。保存したい場所を選んでファイルの名前を入力し、
ダイアログの [OK]ボタンを押すとプログラムがファイルに保存されます。

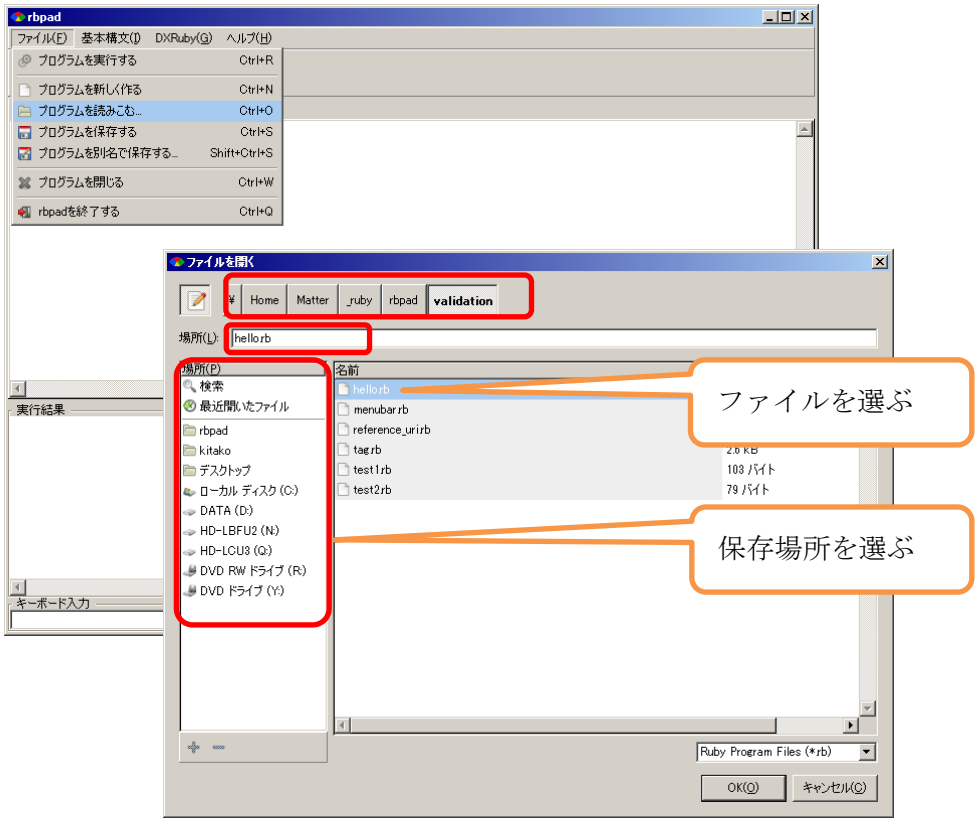


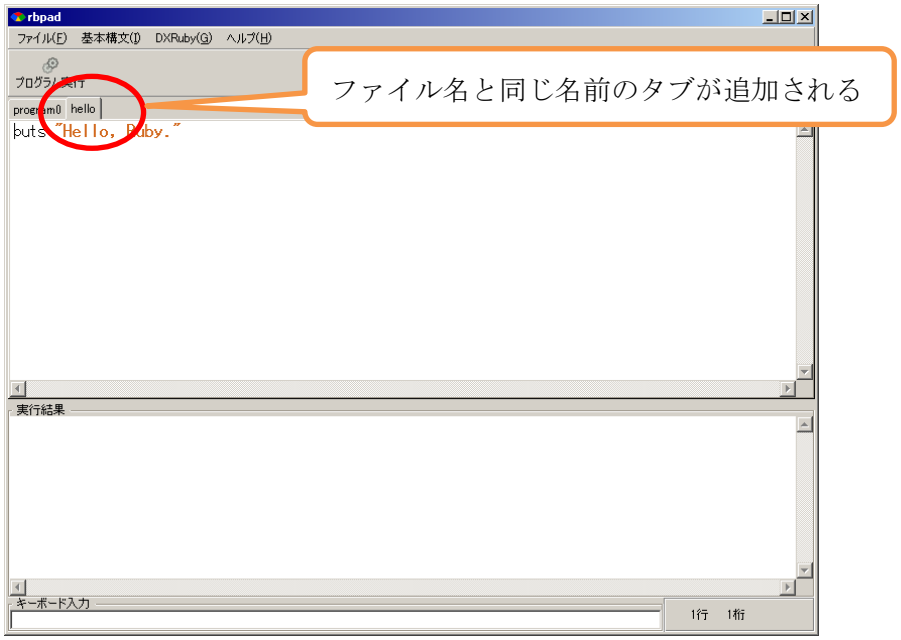
プログラムの保存後はタブにファイル名が表示され、メインウィンドウ右下の「(未保存)」の表示が消えます。



(3)保存したプログラムは、メニューの［ファイル]－[プログラムを読みこむ...]を選び、
ファイル選択ダイアログからファイルを選ぶことによってプログラム編集エリアの
新しいタブに読みこむことができます。
(キーボードで［Ctrl]＋[O]キーを押しても同じ動作になります)

[プログラムを読みこむ...]を選ぶと、読みこむファイルを指定するダイアログが開かれます。
ファイルの場所と名前を選び、ダイアログの［OK]ボタンを押すとプログラムが読みこまれます。



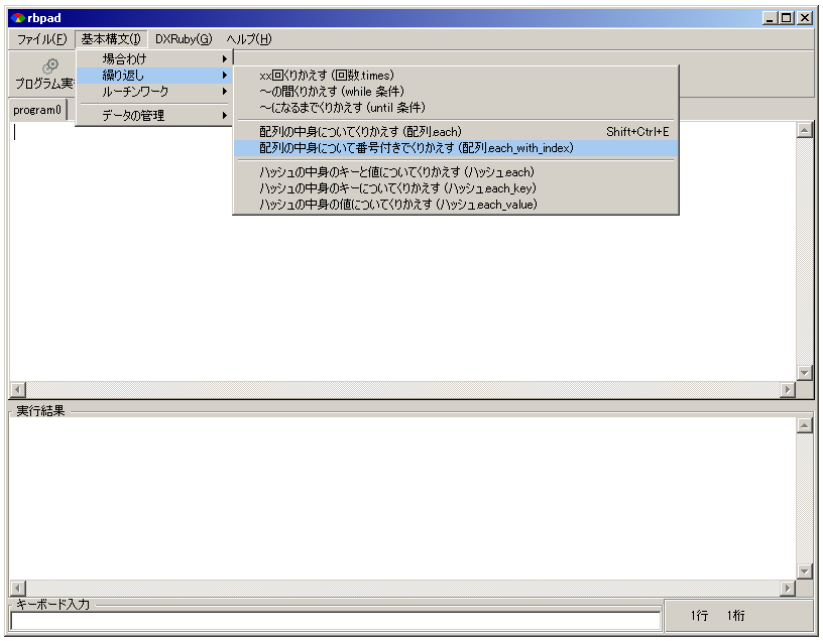


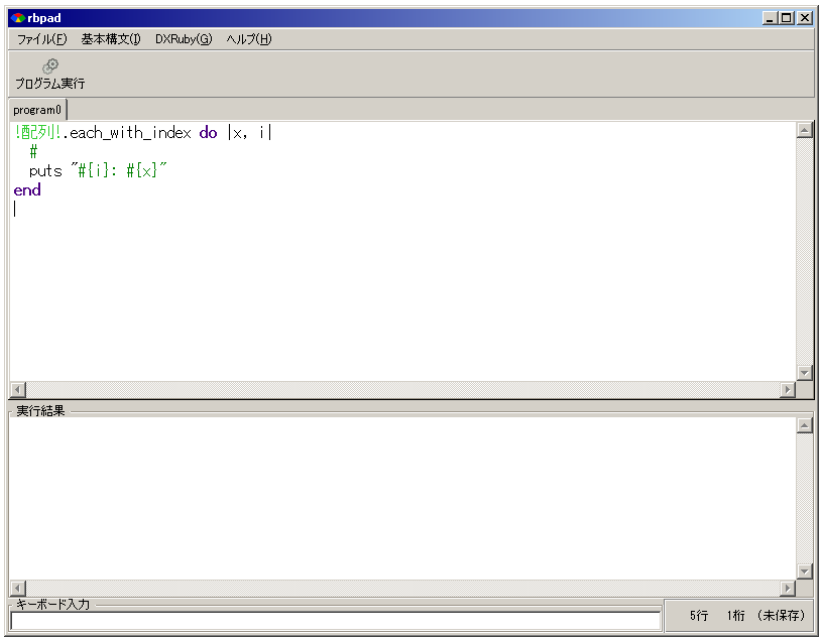
３．基本構文をマスターしていこう

(1)よく使う Ruby の構文を簡単に入力できるように、基本構文メニューの中のサブメニューから基本的な構文を選べるようになっています。

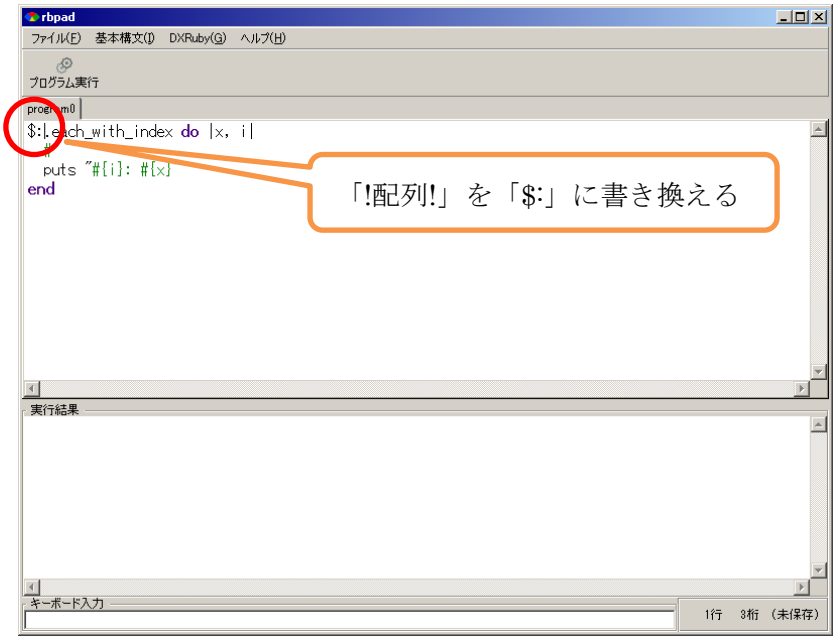
例えば、[基本構文]－[繰り返し]－[配列の中身について番号付きでくりかえす]を選ぶと、下記のような構文がひとかたまりで入力されます。

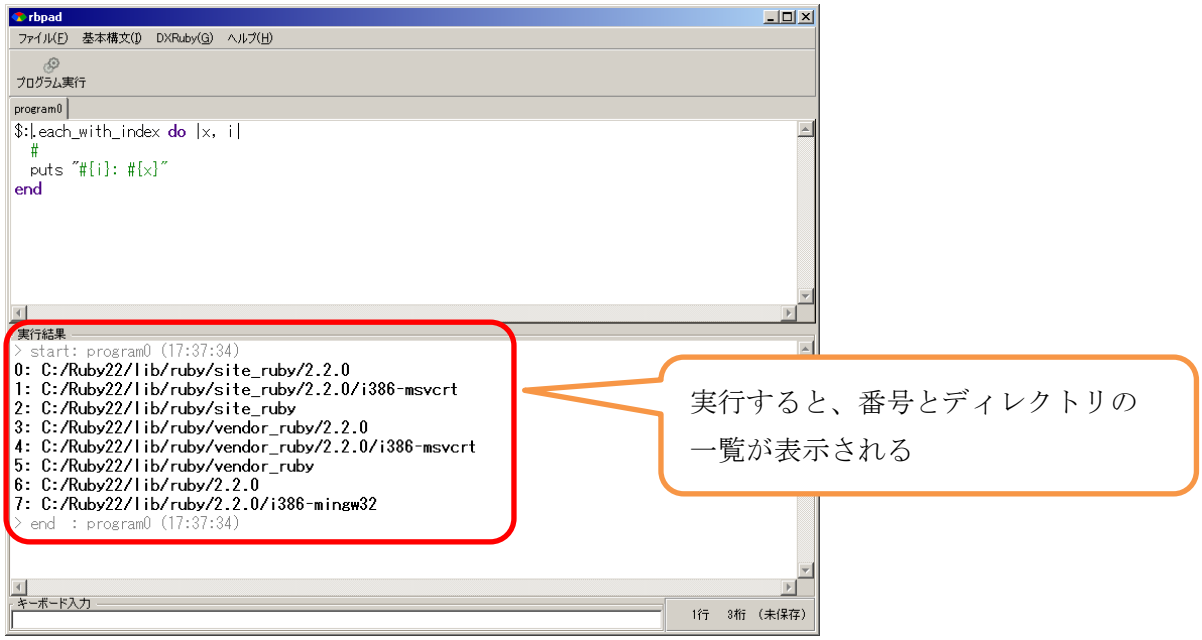
```
!配列!.each_with_index do |x, i|
  #
  puts "#{i}: #{x}"
end
```





薄い緑色で表示される「!配列!」などの部分は、作りたいプログラムにあわせて内容を
書き換えてください。また「#」以降の部分には、コメントや実際におこないたい処理を
書いていってください。





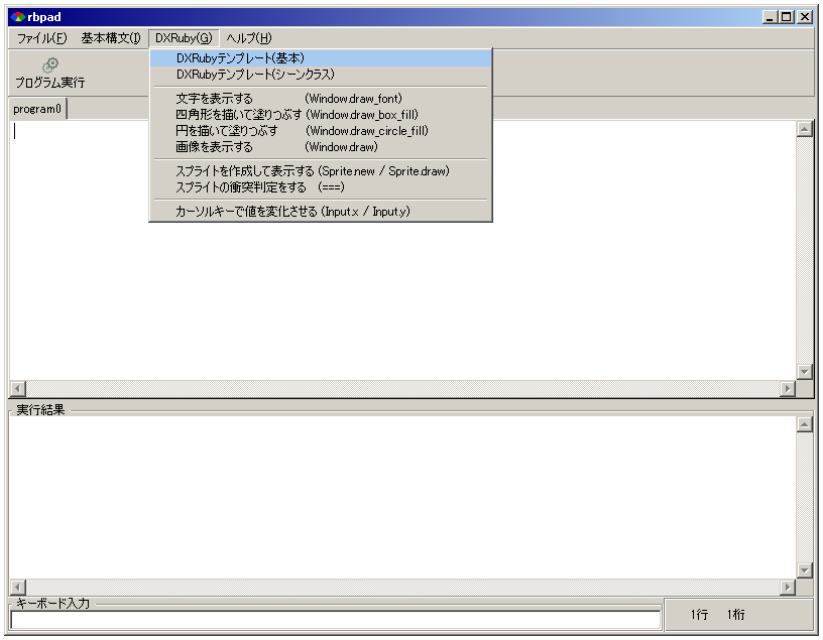
ためしに、上記の「!配列!」の部分を「$:」という記号(特殊変数)に書き換えてからプログラムを
実行してみると、実行結果エリアに番号とディレクトリの一覧が表示されます。
(「$:」には Ruby が必要なファイルを探すときの場所の情報が配列で保持されており、
これはその情報を一覧表示するプログラムになりますが、ここでは詳細な説明は省きます)

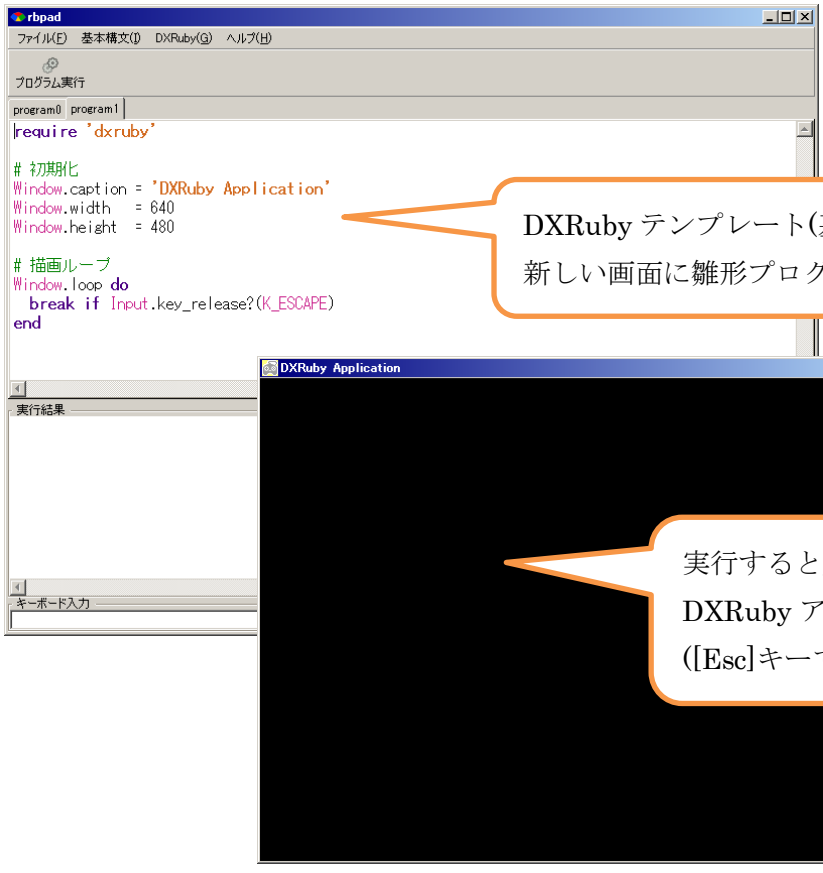
(2)ほかにも、プログラムに必要な基本的な構文がメニューに用意されていますので、
なにかを書こうとしたときに「Ruby ではどういうふうに書くんだっけ？」と迷ったら、
この[基本構文]メニューの中から基本構文を探してみてください。

4．DXRuby テンプレートを使ってみよう

(1)まず、DXRuby のアプリケーションを作るための枠組みを準備します。

メニューの [DXRuby]－[DXRuby テンプレート(基本)]を選ぶと編集エリアに新しいタブが追加され、DXRuby アプリケーションの雛形プログラムが表示されます。





DXRuby テンプレート(基本)を選ぶと、
新しい画面に雛形プログラムが表示される

実行すると真っ黒いだけの
DXRuby アプリケーションが起動する
([Esc]キーで終了)

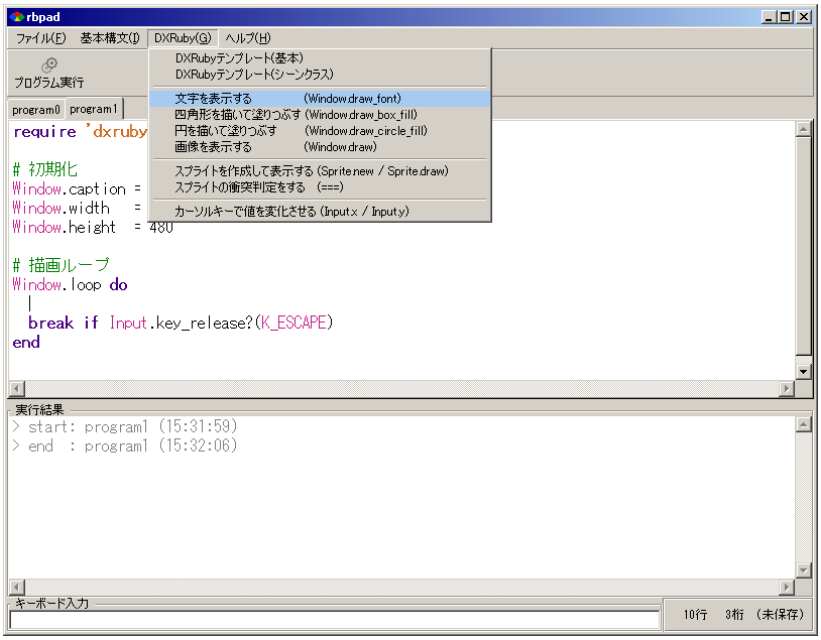
その状態で [プログラム実行]ボタンを押すと、真っ黒い画面が出てきます。
ここには何も表示されていませんが、これも立派な DXRuby のアプリケーションです。

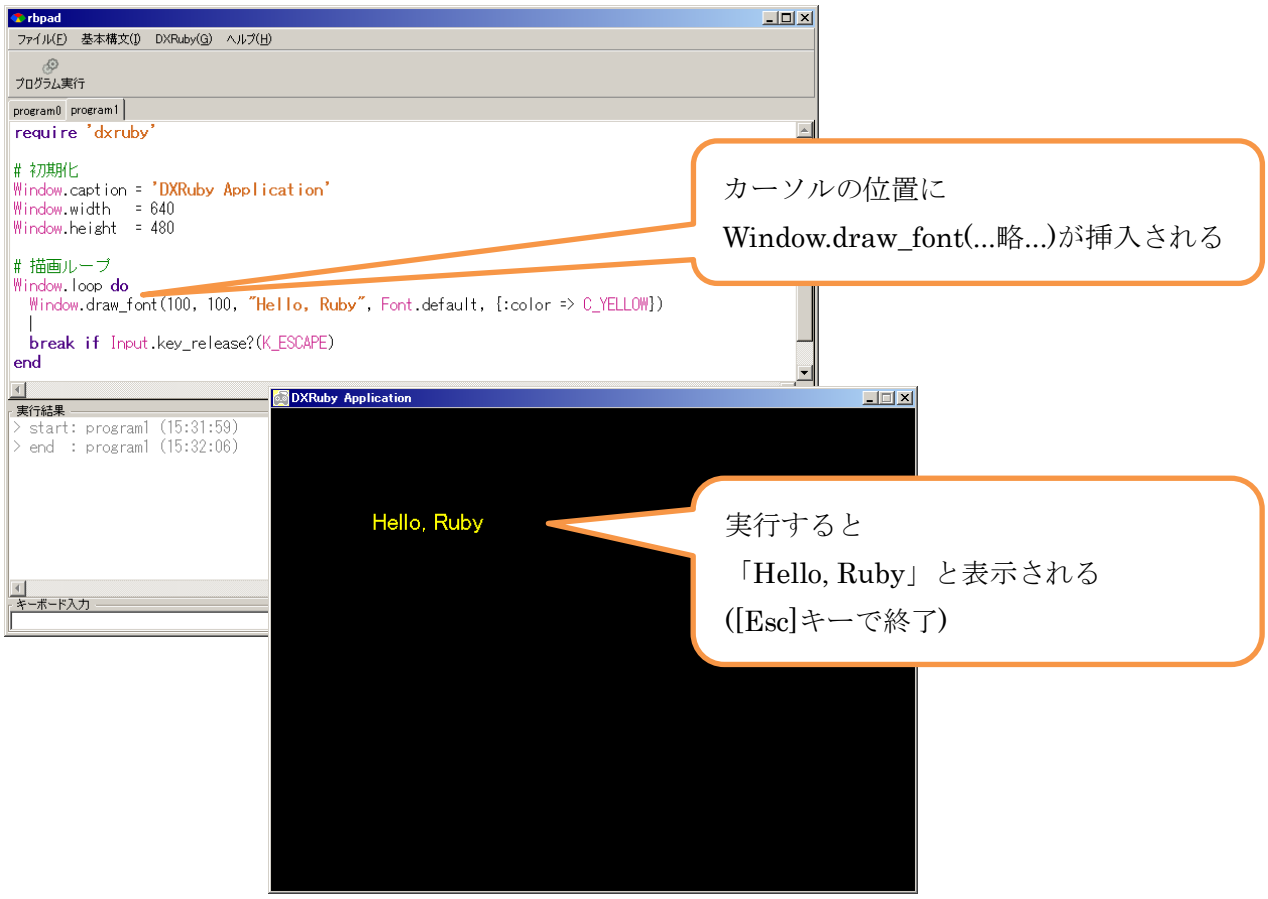
そこまで確認ができたら、キーボードで [Esc]キーを押して DXRuby アプリケーションを終了してください。

(2)つぎに、雛形プログラムに文字の描画を追加してみます。

(1)で表示された雛形プログラムの「Window.loop do」の後ろで改行し、「end」の間に空白行を作ります。

新たにできた空白行の先頭にスペースを２つ入れて、下図の位置にカーソルを移動させてからメニュー [DXRuby]－[文字を表示する]を選んでみてください。



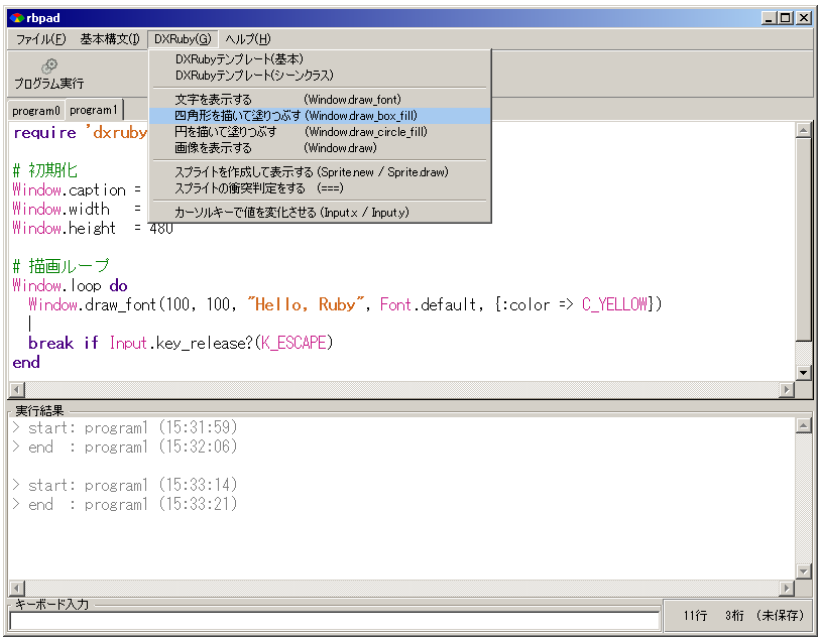


カーソルの位置に「Window.draw_font(...略...)」という１行が挿入されます。
この状態で [プログラム実行]ボタンを押すと、DXRuby アプリケーションの黒い画面の中に「Hello, Ruby」という文字が黄色で表示されます。

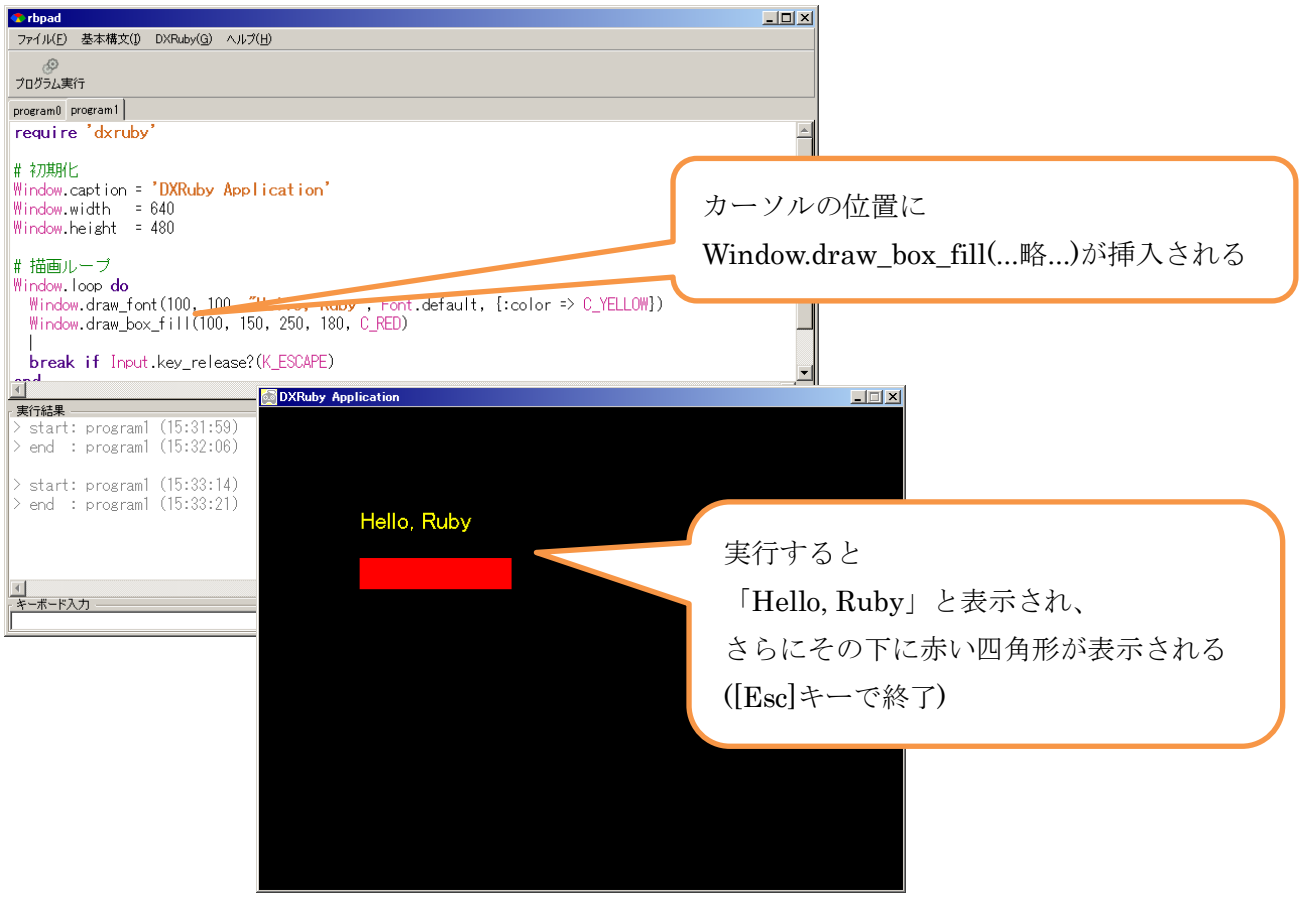
これも、[Esc]キーを押すと DXRuby のアプリケーションを終わらせることができます。

(3)文字の表示のほかに、さらにもう１つメソッドを追加して図形を描いてみます。

先ほどの(2)の編集エリアの状態のまま、メニュー [DXRuby]－[四角形を描いて塗りつぶす]を選んでみてください。



「Window.draw_font(...略...)」の下の行に、今度は「Window.draw_box_fill(...略...)」という
１行が挿入されます。



この状態で[プログラム実行]ボタンを押してみてください。
先ほどの黄色い「Hello, Ruby」の下に、さらに赤い四角形が描画されるようになります。

(4)いろいろ書き換えてみよう

いま(1)～(3)で試したものは、表示する場所や文字や色などがあらかじめ適当に
設定されていましたが、それらはプログラムを書き換えることで自由に変えられます。

自分の好きな場所に、好きな文字や図形が好きな色で描けるよう、いろいろと
試してみてください。

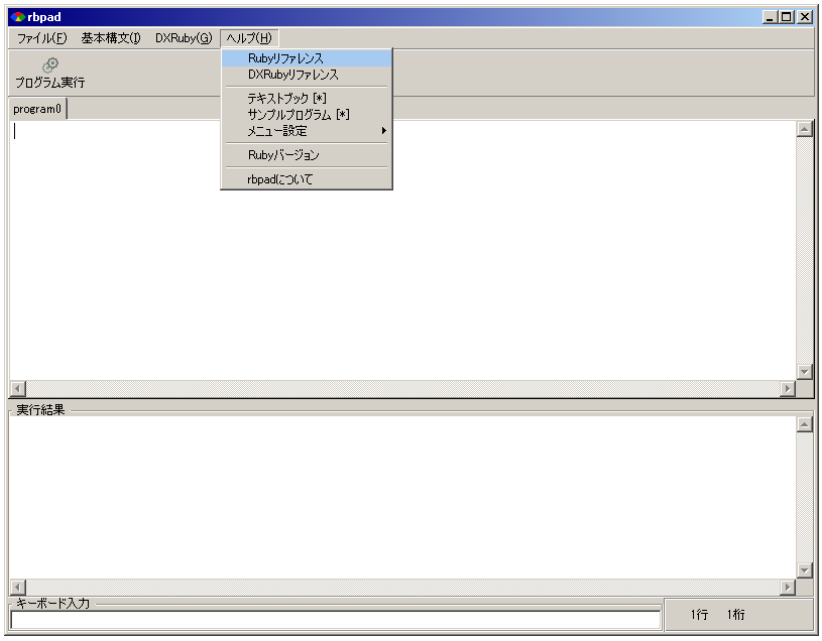
また、メニューの中にはもう少し別のメソッドも用意されていますので、
それらを選ぶと何ができるのかも試してみてください。

５．調べたいときはヘルプを見よう

基本構文や DXRuby のテンプレート機能などは、Ruby の言語仕様の中の ごく基本的なものだけしか載せてありません。

プログラミングをしていて、「こういうときにはなにをどう使えばいいかな？」と疑問に思ったら、面倒がらずにヘルプ(リファレンス)の内容を参照して、いろいろと調べてみてください。

Ruby のヘルプ(WEB サイト)　　...　　[ヘルプ]－[Ruby リファレンス]
DXRuby のヘルプ(WEB サイト) ...　　[ヘルプ]－[DXRuby リファレンス]



そして、ヘルプ(リファレンス)に載っている様々なサンプルコードを rbpad にコピー&ペーストして、実際にどんなふうに動くのか、いろいろと試してみてください。