printemps des sciences

sciences enVies

du 22 au 28 mars 2010

ULB

# Ruby, un Langage de Programmation Pensé Pour ses Utilisateurs

Laurent Peuch
Département d'Informatique

Ruby

*A Programmer's Best Friend*

Ruby est un langage de programmation de très haut niveau, facile d'approche, élégant, moderne et puissant. Il a été créé en 1995 par Yukihiro Matsumoto et ne cesse d'évoluer ; sa dernière version date du mois de décembre. Il permet un apprentissage facile sans pour autant se retrouver limité par la suite. De nombreuses librairies existent, autant pour le web, que pour les jeux ou les interfaces graphiques.

## Un langage pour les humains:

(99 bottles of beer game)

```cpp
#include <iostream>

using namespace std;
template<bool small, int I>
struct pretty_printer;

#define SMALL_PRETTY_PRINTER(num, string) \
template<>\
struct pretty_printer<true, num>\ {\
    static void print()\
    {\
        cout << string;\
    }\
};

SMALL_PRETTY_PRINTER(0, "No")
SMALL_PRETTY_PRINTER(1, "One")
SMALL_PRETTY_PRINTER(2, "Two")
SMALL_PRETTY_PRINTER(3, "Three")
SMALL_PRETTY_PRINTER(4, "Four")
SMALL_PRETTY_PRINTER(5, "Five")
SMALL_PRETTY_PRINTER(6, "Six")
SMALL_PRETTY_PRINTER(7, "Seven")
SMALL_PRETTY_PRINTER(8, "Eight")
SMALL_PRETTY_PRINTER(9, "Nine")
SMALL_PRETTY_PRINTER(10, "Ten")
SMALL_PRETTY_PRINTER(11, "Eleven")
SMALL_PRETTY_PRINTER(12, "Twelve")
SMALL_PRETTY_PRINTER(13, "Thirteen")
SMALL_PRETTY_PRINTER(14, "Fourteen")
SMALL_PRETTY_PRINTER(15, "Fifteen")
SMALL_PRETTY_PRINTER(16, "Sixteen")
SMALL_PRETTY_PRINTER(17, "Seventeen")
SMALL_PRETTY_PRINTER(18, "Eighteen")
SMALL_PRETTY_PRINTER(19, "Nineteen")

#undef SMALL_PRETTY_PRINTER
template<int ones>
inline void
print_ones();

#define ONES_PRINTER(ones, string) \
template<> \
inline void \
print_ones<ones>() \ {\
  cout << string;\
}

ONES_PRINTER(0, " ")
ONES_PRINTER(1, " one")
ONES_PRINTER(2, " two")
ONES_PRINTER(3, " three")
ONES_PRINTER(4, " four")
ONES_PRINTER(5, " five")
ONES_PRINTER(6, " six")
ONES_PRINTER(7, " seven")
ONES_PRINTER(8, " eight")
ONES_PRINTER(9, " nine")

#undef ONES_PRINTER
template<int tens>
inline void
print_tens();

#define TENS_PRINTER(tens, string) \
template<> \
inline void \
print_tens<tens>() \ {\
  cout << string;\
}

TENS_PRINTER(2, "Twenty")
TENS_PRINTER(3, "Thirty")
TENS_PRINTER(4, "Forty")
TENS_PRINTER(5, "Fifty")
TENS_PRINTER(6, "Sixty")
TENS_PRINTER(7, "Seventy")
TENS_PRINTER(8, "Eighty")
TENS_PRINTER(9, "Ninety")

#undef TENS_PRINTER
template<int I>
struct pretty_printer<false, I> {
    static void print(){
        print_tens<(I - I%10)/10>();
        print_ones<(I%10)>();
    }
};

template<int I>
void pretty_print() {
    pretty_printer<(I<20), I>::print();
}

template<int I>
inline void
BottlesOfBeer() {
    pretty_print<I>();
    cout << " bottles of beer" ;
}

template<>
inline void
BottlesOfBeer<1>() {
    pretty_print<1>();
    cout << " bottle of beer" ;
}

template<int I>
inline void
BottlesOfBeerOnTheWall() {
    BottlesOfBeer<I>();
    cout << " on the wall";
}

template<int I>
inline void stanza() {
    BottlesOfBeerOnTheWall<I>();
    cout << ",\n";
    BottlesOfBeer<I>();
    cout <<",\n";
}

template<int I>
inline void bridge() {
    cout   << "Take one down, pass it around," << endl;
    BottlesOfBeerOnTheWall<I-1>();
    cout <<",\n";
}

template<>
inline void bridge<0>() {
    cout << "Go to the store and buy some more," << endl;
    BottlesOfBeerOnTheWall<99>();
}

template<int I>
inline void verse() {
    stanza<I>();
    bridge<I>();
}

template<int I>
inline void sing () {
    verse<I>();
    cout << endl
    sing<I-1>();
}

template<>
inline void sing<0> () {
    verse<0>();
}

int main () {
  sing<99>();
}
```

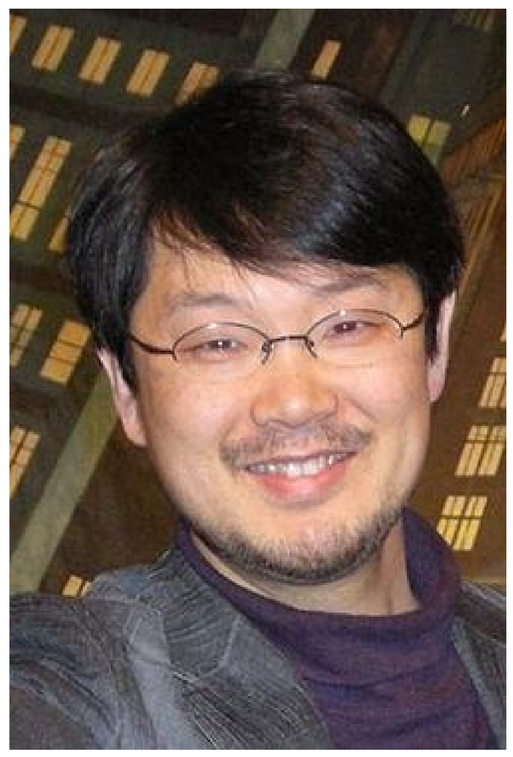
C++

```ruby
#!/usr/bin/env ruby
# -*- encoding: utf-8 -*-

class Integer # The bottles
  def drink; self - 1; end
end

class << song = nil
  attr_accessor :wall

  def bottles
    (@bottles.zero? ? "no more" : @bottles).to_s <<
      " bottle" << ("s" unless @bottles == 1).to_s
  end

  def of(bottles)
    @bottles = bottles
    (class << self; self; end).module_eval do
      define_method(:buy) { bottles }
    end
    self
  end

  def sing(&step)
    puts "#{bottles.capitalize} of beer on the wall, #{bottles} of beer."
    if @bottles.zero?
      print "Go to the store buy some more, "
      step = method :buy
    else
      print "Take one down and pass it around, "
    end
    @bottles = step[@bottles]
    puts "#{bottles} of beer on the wall."
    puts "" or wall.call unless step.kind_of? Method
  end

end

callcc { |song.wall| song.of(99) }.sing { |beer| beer.drink }
```

Ruby

## Caractéristiques:

- une syntaxe simple et épurée,
- un langage très haut niveau, plus facilement abordable et plus puissant,
- un typage dynamique pour une plus grande simplicité,
- pensé pour le confort de son utilisateur et pour être élégant,
- mutiplateforme, votre code fonctionnera partout,
- un ramasse-miettes pour ne pas avoir à se soucier de la mémoire,
- intégration des expressions régulières directement dans le langage,
- complètement orienté objet,
- une librairie standard très complète ,
- ruby on rails,
- multi-paradigme (impératif, objet, fonctionnel, ...),
- des fonctions avancées puissantes: itérateurs, générateurs, ranges, blocs de code ...



Yukihiro Matsumoto

(Photo: wikipedia)

FACULTE DES SCIENCES
UNIVERSITE LIBRE DE BRUXELLES