Динамическое программирование

Информатика
10-11 классы

17 ноября 2011 г.

Разбор задач: средняя длина слова

- Найдём среднюю длину слова в три этапа:
  1. Вычислим количество слов в строке.
  2. Найдём длину всей строки без пробелов (сумма длин слов).
  3. Поделим одно на другое.

Listing 1: Средняя длина слова

```
s = "This is a sample string"
words = s.split(" ")
size = words.size
res = 0.0
words.each do |elem|
  res = res + elem.size
end
medium = res/size
puts medium
```

Разбор задач: слова, длиннее следнего

- Найдём среднюю длину строки в два этапа:
  1. Посчитаем среднюю длину слова.
  2. Найдём слова, чья длина больше средней.
- ... пропустим кусок, связанный с п.1.

Listing 2: Длиннее среднего

```
...
medium = res/size
words.each do |elem|
  puts elem if (elem.size > medium)
end
```

Динамическое программирование: принципы

- Решая предыдущие задачи, мы столкнулись с тем, что:
  1. Очень удобно разбивать задачу над несколько небольших подзадач.
  2. Зачастую, одна программа использует результаты, полученные другой программой.
  3. А в некоторых случаях подзадачи нескольких задач совпадают.
- Такой подход называется *динамическим программированием*.
- *Динамическое программирование* — способ решения сложных задач путём разбиения их на более простые подзадачи (Википедия).

Инструменты

- Для разбиения программы на подпрограммы в языках программирования существует специальный инструмент, называемый *функциями*.
- В общем случае, *функция* — это небольшая программа, реализующая парадигму *чёрного ящика*.
- *Парадигма чёрного ящика* предлагает разделение сущности на составляющие. Каждая составляющая имеет *вход*, сам, собственно, *чёрный ящик* и *выход*.
- Чёрный ящик — универсальная концепция, применимая не только в программировании, но и в жизни.

Пример чёрного ящика

Рис.: Источник: http://g0l.ru/blog/imgs/gai/caricatura.jpg

Функции

- Как уже было указано, простейшей реализацией чёрного ящика является функция.
- Функция — подпрограмма внутри программы, выполняющая некоторую функцию и имеющая **входные данные**, **тело (код) функции**, **выходные данные** (что функция возвращает)
- Приведём простейший пример функции: перевод из градусов Цельсия в градусы по Кельвину.

Listing 3: Цельсии в Кельвины

```
def celsius_to_kelvin(cels_value)
  kelv_value = cels_value + 273.15
  return kelv_value
end

sample = celsius_to_kelvin(20)
puts "20 by Celsius = #{sample} by Kelvin"
```

Функция в разрезе

- **def** название_функции(аргументы)
- В начале задания функции пишется ключевое слово **def** (англ. *definition* — определение).
- Затем пишется название функции. Оно должно понятным и на английском языке без пробелов и спецсимволов.
- Далее, в скобках перечисляются входные данные, которые называются *аргументы функции*.
- Аргументы функции — это несколько переменных, которые нужны функции для её вычислений.
- В конце функции пишется ключевое слово **return** (англ. *возврат*). После return указывается, что возвращает функция.
- В самом конце ставится **end**.
- В нашем случае функция возвращала значение температуры в градусах по Кельвину.

Функции для чайников

Рис.: Источник: http://www.dialektika.com/

Ещё немного о функциях

- Для вызова функции достаточно написать её имя, а затем перечислить в скобках аргументы.
- Например *puts celsius_to_kelvin*(20) выведет на экран значение 20 градусов цельсия по шкале Кельвина.
- Можно и через дополнительную переменную:

Listing 4: Функции

```
...
celsius_value = 20
in_kelvin = celsius_to_kelvin(celsius_value)
puts in_kelvin
```

И ещё чуть-чуть

Рис.: Источник: где-то в Интернетах

И ещё чуть-чуть

- **Важно!** Все переменные внутри функции *локальные*. То есть, если в программе есть переменная **res**, внутри функции её **НЕ БУДЕТ**. И наоборот.
- Такая программа вызовет ошибку, так как переменная внутри функции не определена:

Listing 5: Ошибка с локальными переменными

```
elem = 5
def very_useful_function(something)
  puts elem
  puts something
  return elem
end
```

Задача о палиндромоме

- *Палиндромом* называют слово (или буквосочетание), одинаково читающееся в обоих направлениях: топот, *А роза упала на лапу Азора* (Фет).
- Задача: вывести на экран все палиндромы–слова, встречающиеся в строке.
- Решим задачу методом динамического программирования:

1. Разделим предложение на слова.
2. Проверим каждое слово, является ли оно палиндромом (функция).
3. Если является, то выведем его на экран.

Задача о палиндромоме

Listing 6: Задача о палиндроме

```
def palindrome?(s)
  if (s == s.reverse)
    return true
  else
    return false
  end
end

s = "I said to madam after party: WOW"
words = s.split(" ")
words.each {|word| puts word if palindrome?(word) }
```

Числа Фибоначчи

- Выведем на экран первые N чисел Фибоначчи.

Listing 7: Числа Фибоначчи

```
def fibonacci(n)
  a0 = 1
  a1 = 1
  a_new = 0
  for i in 2..n
    a_new = a0 + a1
    a0 = a1
    a1 = a_new
  end
  return a1
end
n = 5
for i in 0..n
  puts "#{i} число Фибоначчи = #{fibonacci(i)}"
end
```

References

- Все презентации доступны на http://school.smirik.ru!
- Вопросы, предложения, д/з: smirik@gmail.com