Разбор итоговых задач

Информатика
10-11 классы

23 апреля 2012 г.

Задача 1

- Дан целочисленный массив из 30 элементов. Элементы могут принимать значения от 0 до 100 – баллы, полученные на ЕГЭ. Опишите на русском языке или на одном из языков программирования алгоритм, который подсчитывает и выводит средний балл учащихся, сдавших экзамен (получивших оценку более 20 баллов). Гарантируется, что хотя бы один ученик в классе успешно сдал экзамен.

Пример

Поиск элементов

- В ряде задач нам нужно извлечь из массива определённые элементы.
- Допустим дана средняя температура за 5 дней: arr = [5.3, 2.1, 1.2, -0.8, -0.2].
- Вычислим, сколько дней была отрицательная температура.

Listing 6: Метод find_all

```
arr  = [5.3, 2.1, 1.2, -0.8, -0.2]
arr_neg = arr.find_all{|elem| (elem < 0) }
puts arr_neg.size

puts [5.3, 2.1, 1.2, -0.8, -0.2].
     find_all{|elem| (elem > 0) }.size
```

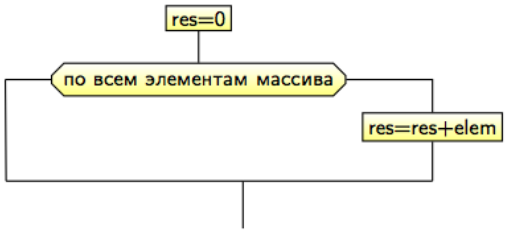
Пример

Сумма

- Допустим дана средняя температура за 5 дней: arr = [5.3, 2.1, 5.2, 1.8, -0.2].
- Как вычислить сумму?

Listing 4: Сумма

```
arr = [5.3, 2.1, 5.2, 1.8, -0.2]
sum = arr.inject(0){ |res, elem| res+elem }
```



Презентация №4 — массивы

Решение задачи 1

Listing 1: Решение задачи 1

```
arr = [20, 87, 23, 53, 100, 2]
passed = arr.find_all{ |elem| elem > 20 }
sum_passed = passed.inject(0){ |res, elem| res+elem }
average = sum_passed / passed.size
puts average
```

- Задаём исходные данные.
- Находим тех, кто сдал + сумму их баллов.
- Делим сумму на количество и выводим результат.

Диофантовы уравнения

- Напишите эффективную программу, находящую все решения линейного диофантова уравнения $ax + by = c$,
- где a, b, c — натуральные числа, задающиеся в начале программы, x, y — неизвестные (**целые числа**).
- Ограничение: $|x|, |y| < 100$; то есть надо найти решения, где оба неизвестных не превосходят 100.

Метод перебора

Метод перебора

- Идея метода перебора очень проста — переберём все варианты и выберем только те, которые нам нужны.
- Рассмотрим простую задачу: вывести на экран простые числа <= N.
- Есть красивые решения типа решета Эратосфена. Но мы пойдём “в лоб”.

Listing 1: Простые числа

```
def is_prime?(n)
  return false if ((n == 1) || (n == 0))
  for i in 2..(n-1)
    return false if (n%i == 0)
  end
  true
end

100.times {|i| puts i if is_prime?(i) }
```


Презентация №9 — перебор и рекурсия

Алгоритм решения задачи 4

1. Переберём все возможные пары (x, y) от -100 до 100.
2. Подставим каждую из них в уравнение $ax + by = c$.
3. Если полученное равенство будет истинным, выведем на экран значения (x, y).

Решение задачи 4

Listing 2: Задача 4

```
a = 12
b = 16
c = 100
for x in -100..100
  for y in -100..100
    puts "x = #{x}, y = #{y}" if (a*x + b*y == c)
  end
end
```

Задача 2

- Напишите программу, подсчитывающую максимальное количество подряд идущих отрицательных элементов в целочисленном массиве длины 30.
- **Решение**.
- 1, -2, -5, 9, 1, -4, 3, 0, -5, -1, -3, -4, 0, -1
- Ответ: 4. Давайте посмотрим, как мы в уме находим это количество?
- Мы просматриваем все числа слева направо. Ищем первое отрицательное.
- Далее, считаем количество отрицательных чисел, идущих после первого.
- Дойдя до положительного числа, запоминаем, сколько пока было максимально подряд идущих элементов.

Задача 2. Решение.

- 1, -2, -5, 9, 1, -4, 3, 0, -5, -1, -3, -4, 0, -1
- На примере: идём до числа -2.
- Считаем количество идущих за ним отрицательных элементов. Их будет 2. Запоминаем это число.
- Пропускаем 9 и 1, так как они — положительные.
- Берём -4. Считаем количество идущих после -4 отрицательных чисел. Их нет. Значит, в данной группе всего 1 число. Ранее было 2.
- Значит, максимальное число подряд идущих отрицательных элементов так и остаётся равным 2.
- Идём далее. 0 пропускаем, доходим до -5. Там — 4 подряд идущих отрицательных элемента.
- $4 > 2$, поэтому запоминаем число 4, как текущий максимум. Продолжаем так до конца.

Формализация

- Итого, что нам надо две переменных:
  1. Переменная, в которой мы будем хранить текущее найденное максимальное количество подряд идущих элементов массива (max).
  2. Переменная, в которой мы будем хранить количество подряд идущих отрицательных элементов в одной группе (current).

Число	1	-2	-5	9	1	-4	3	0	-5	-1	-3	-4	0	-1
current	0	1	2	0	0	1	0	0	1	2	3	4	0	1
max	0	1	2	2	2	2	2	2	2	2	3	4	4	4

Формализация условий

- Формализуем условия, при которых изменяются переменные *current* и *max*.
- Если пробегаемое число — отрицательное, то увеличиваем current на единицу:
- current += 1 if elem < 0.
- Если пробегаемое число больше или равно нулю, то обнуляем current:
- current = 0 if elem >= 0.
- Если количество элементов в группе превысило максимальное, изменяем последнее:
- max = current if current > max.

Программа для задачи 2

Listing 3: Задача 2

```
arr = [1, -2, -5, 9, 1, -4, 3, 0, -5, -1, -3, -4, 0, -1]
current = 0
max = 0
for i in 0..arr.size-1
  current += 1 if arr[i] < 0
  current = 0  if arr[i] >= 0
  max = current if current > max
end
puts max
```

Альтернативное решение

Listing 4: Альтернативное решение задачи 2

```
arr = [1, -2, -5, 9, 1, -4, 3, 0, -5, -1, -3, -4, 0, -1]
current = 0
max = 0
arr.each do |elem|
  current += 1 if elem < 0
  current = 0  if elem >= 0
  max = current if current > max
end
puts max
```

References

- При подготовке данного материала использовались сайты: http://ru.wikibooks.org/wiki/Ruby, http://rubydev.ru, http://en.wikipedia.org, http://ruby-lang.org.
- Все презентации доступны на http://school.smirik.ru!
- Вопросы, предложения, д/з: smirik@gmail.com