Jeremy Diamond PA1

PA1 asks us to build a program that reads in several integers from a txt file, find the mean of the integers, and find the number of integers that are of a greater magnitude than the average. This program was to be implemented in the language Julia.

Approach and implementation details

I split the problem into 3 functions. First a read in function to handle the file format, second a function to find the mean and a third to count the number of values greater than the mean. The read\_in function is modified from the example code provided with 3 changes. I eliminated the truple structure to simplify it as that functionality was not needed for this implementation, I simply told the read in to skip the first 2 lines to meet the file format provided in one\_million\_randoms.txt and I made it print the size for debugging reasons. The mean function takes in an array of int 64 type and prints/returns a float 64 that is the mean of all the numbers in the array. The findNumAbove function takes in both the array of integers and the mean. It then increments a counter for every integer that is greater than the mean. It then returns that count and the program prints it. Notably I used julia’s built in length() operation to avoid storing the size of the array (This aint assembly. I’m not manually tracking sizes.)

Code

function read\_array(name::String)

data = Int64[]

daCount = 0

open(name) do file

for line in eachline(file)

if daCount >= 2 #skips first 2 lines

push!(data,parse(Int64,line))

end

daCount += 1

end

end

#for debugging

print("size is ")

println(length(data))

return data

end

function mean(data::Array{Int64})

sum::Int64 = 0

#sums the numbers

for i = 1:length(data)

sum += data[i]

end

#for debugging

print("total sum is ")

println(sum)

#type convertion to float happens automatically

meanVal = sum/length(data)

#for debugging

print("meanVal is ")

println(meanVal)

return meanVal

end

function findNumAbove(data::Array{Int64},meanVal::Float64)

daCount = 0

for i = 1:length(data)

if data[i] > meanVal

daCount += 1

end

end

return daCount

end

#change string for differant files

data = read\_array("C:\\Users\\Jeremy G Diamond\\Documents\\principals\\randomtest2.txt")

meanVal = mean(data)

allDaNumbs = findNumAbove(data,meanVal)

print("The number of entries above the mean is ")

println(allDaNumbs)

Sample runs

My samples are all versions of the one million randoms file but different sections with the first 2 lines attached. These extra files will be copied at the end

Randomtest1.txt results

size is 23

total sum is 9155

meanVal is 398.04347826086956

The number of entries above the mean is 10

Randomtest2.txt results

size is 108

total sum is 55140

meanVal is 510.55555555555554

The number of entries above the mean is 58

One\_million\_randoms.txt

size is 1000000

total sum is 499021498

meanVal is 499.021498

The number of entries above the mean is 499408

Conclusion

Julia is a promising language with some early growing pains. The rich class structure is nice but leaves room for a lot of mistakes. The python like lack of a main is cool but it should allow you to type variables. The IDE packages available are clean but lack features. And the compiler errors are useless. I’m going to keep an eye on this to see how it develops in the future.

Appendix

Txt files

Randomtest1.txt

One Million Random Integers 0..999

1000000

581

930

391

852

509

215

262

292

80

832

336

260

727

32

58

255

157

554

444

628

10

23

727

Randomtest2.txt

One Million Random Integers 0..999

1000000

830

528

758

278

164

551

318

666

507

850

82

447

502

101

494

396

457

978

860

722

70

514

39

151

40

592

699

462

572

766

589

857

719

199

81

855

512

893

425

181

125

744

186

404

864

880

542

788

592

496

452

664

532

580

172

725

284

529

185

43

669

185

333

248

306

194

19

258

563

311

769

595

688

294

351

681

725

811

505

482

705

971

833

289

34

119

913

342

959

644

877

915

864

576

100

521

92

842

547

814

739

313

674

643

464

615

478

778