**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

Данную лабораторную работу я выполнил на BASH. Для выполнения данной лабораторной работы были написаны три скрипта: один для расчета оценки числа π, второй для сравнения точности вычислений в зависимости от количества бросков, третий для анализа ГСЧ на плотность распределения.

Данная лабораторная работа разделена на несколько пунктов:

1)Создание выборки с помощью генератора случайных чисел

2)Проверка попаданий полученных точек в окружность

3)Вычисление числа π

4)Обработка полученных данных

5)Анализ рандомайзера на плотность распределения

**КОД**

Код первого скрипта «1st-lab.sh» представлен ниже:

#!/bin/bash #это шебанг, тут указываем путь к оболочке которой мы должны скормить данный код

RAND\_MAX=32767 #задаем максимальное значение которое может принять системная переменная

#$RANDOM, которая генерирует случайное число от 0 до 32767

random\_in\_range () {

local R=$1 #создаем локальную переменную радиуса, которой присваевается значение первого аргумента переданного в данную функци

echo $(bc <<< "scale=5; $R \* ( $RANDOM / $RAND\_MAX )") #bc - basic calculator, утилита линукс представляющая из себя калькулятор

#тут я передаю в утилиту в качестве входных данных строку "scale=5; $R \* ( $RANDOM / $RAND\_MAX) "

#с помощью оператора <<< который создает временный файл в который записывает нашу строку, после чего утилита считывает этот файл

#и использует данные из него как входные

#scale=5 - количество знаков после запятой, $R \* ( $RANDOM / $RAND\_MAX) - генерация случайного значения от 0 до R

}

calc\_pi () {

local R=$1 #создаем локальную переменную радиуса, которой присваевается значение первой переменной переданной в функцию

local N=$2 #создаем локальную переменную количества бросков

local hits=0 #создаем локальную переменную количества попаданий

for((i=0; i<=N; i++))

do

x=$(random\_in\_range "$R") #создаем переменную x в которую записываем случайное значение полученное из функции random\_in\_range

y=$(random\_in\_range "$R") #создаем переменную y в которую записываем случайное значение полученное из функции random\_in\_range

dist=$(bc <<< "scale=5; ($x^2 + $y^2) <= $R^2") #переменная в которой проверяем попали ли координаты или нет

if [[ "$dist" = 1 ]] # проверка

then

hits=$(( $hits + 1 ))

fi

done

pi\_estimate=$(bc <<< "scale=5; 4 \* $hits / $N") #оценка ПИ

echo "the estimate of pi number: $pi\_estimate" #выводим оценку

}

throws=(10 100 1000 10000) #создаем массив в который закидываем количество бросков

radius=1 #радиус

total\_exec\_time=0 #переменная для подсчета общего времени исполнения скрипта

for N in "${throws[@]}" #foreach цикл, синтаксическая конструкция ${throws[@]} разворачивается во все значения которые есть в массиве throws

do

echo "Number of throws: $N" >> output.txt #выводим в файл

for j in {1..5} #foreach цикл, {1..5..increment} - разворачивается в последовательность 1 2 3 4 5, тк инкремент не указан он по умолчанию 1

do

start\_time=$(date +%s%N) #время начала работы

calc\_pi "$radius" "$N" >> output.txt #записываем результат в файл

end\_time=$(date +%s%N) #время конца работы

duration=$(( end\_time - start\_time )) #время работы

total\_exec\_time=$(( $total\_exec\_time + $duration )) #общее время работы

echo "Execution time for $N throws $j: $(( $duration / 1000000000 )) sec, $(( ($duration / 1000000) % 1000 )) ms" >> exec-time.txt

#время работы записываем в файл

done

echo

done

echo "Total exectuion time: $(( $total\_exec\_time / 1000000000 )) sec, $(( ($total\_exec\_time / 1000000) % 1000 )) ms"

Код второго скрипта «compare-accuracy.sh» представлен ниже:

#!/bin/bash

#PI\_NUMBER=3.14159

file\_input="output.txt" #название файла из которого берем данные

file\_output="accuracy.txt" #название файла в который выводим результат

declare -A pi\_sums #объявляет под именем pi\_sums ассоциативный массив

declare -A pi\_counts #объявляет под именем pi\_counts ассоциативный массив

calculate\_avg () { #тут и так все понятно, имхо

for throw in "${!pi\_sums[@]}"

do

sum=${pi\_sums[$throw]}

count=${pi\_counts[$throw]}

avg=$(bc <<< "scale=5; $sum / $count")

echo "Number of throws: $throw, average estimate of pi: $avg" # >> "$file\_output"

done | sort -r >> "$file\_output" #прежде чем вывести в файл, передаем наши данные в утилиту sort

#с ключом -r который ответственен за сортировку в обратном порядке после чего результат выводим в файл

}

current\_throws=0

while read -r line #читаем файл

do

if [[ $line == "Number of throws:"\* ]] #и так все понятно

then

current\_throws=$(echo $line | awk '{print $4}') #с помощью утилиты awk получаем количество бросков, которое является 4-м аргументом в строке

pi\_sums[$current\_throws]=0 #все просто

pi\_count[$current\_throws]=0 #все просто

fi

if [[ $line == "the estimate of pi number:"\* ]] #все понятно

then

pi\_val=$(echo $line | awk '{print $6}') #с помощью утилиты получаем оценку числа пи, которая является 6 аргументом в строке

pi\_sums[$current\_throws]=$(bc <<< "${pi\_sums[$current\_throws]} + $pi\_val") #все просто

pi\_counts[$current\_throws]=$(( ${pi\_counts[$current\_throws]} + 1 )) #все просто

fi

done < "$file\_input" #считываем из файла

calculate\_avg

Код третьего скрипта «analysis-rng.sh» представлен ниже:

#!/bin/bash

RAND\_MAX=32767 #уже было

generate\_hundred\_numbers () { #генерируем 100 чисел от 0 до 1

for i in {1..100}

do

echo $( bc <<< "scale=5; $RANDOM / $RAND\_MAX" )

done

}

calc\_stats () {

local numbers=("$@") #записываем все значения переданные в функцию в массив

local sum=0

local sum\_sq=0

local len=${#numbers[@]} #размер массива

for i in "${numbers[@]}"

do

sum=$(bc <<< "$sum + $i")

sum\_sq=$(bc <<< "$sum\_sq + $i^2")

done

avg=$(bc <<< "scale=5; $sum / $len")#мат ожидание

disp=$(bc <<< "scale=5; ($sum\_sq / $len) - ($avg^2)")#дисперсия

echo -e "Math expectation: $avg\nDispersion: $disp" >> result-analysis.txt

}

numbers=($(generate\_hundred\_numbers)) #генерируем 100 чисел записывая их в массив

calc\_stats "${numbers[@]}" #вызываем функцию передавая туда все значения из массива

**Результат**

Оценка числа π:

Number of throws: 10

the estimate of pi number: 4.00000

the estimate of pi number: 2.40000

the estimate of pi number: 3.20000

the estimate of pi number: 4.00000

the estimate of pi number: 3.60000

Number of throws: 100

the estimate of pi number: 3.28000

the estimate of pi number: 3.04000

the estimate of pi number: 3.12000

the estimate of pi number: 3.44000

the estimate of pi number: 3.32000

Number of throws: 1000

the estimate of pi number: 3.16000

the estimate of pi number: 3.21200

the estimate of pi number: 3.15600

the estimate of pi number: 3.21200

the estimate of pi number: 3.07200

Number of throws: 10000

the estimate of pi number: 3.10880

the estimate of pi number: 3.17320

the estimate of pi number: 3.15800

the estimate of pi number: 3.13800

the estimate of pi number: 3.15840

Точность вычисления в зависимости от количества бросков:

Number of throws: 10, average estimate of pi: 3.44000

Number of throws: 100, average estimate of pi: 3.24000

Number of throws: 1000, average estimate of pi: 3.16240

Number of throws: 10000, average estimate of pi: 3.14728

Анализ рандомайзера на плотность распределения:

Math expectation: 0.51389

Dispersion: 0.08090

Математическое ожидание для равномерного распределения определяется по следующей формуле (рис. 1).:

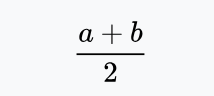


Рисунок 1 — Формула математического ожидания

Соответственно, математическое ожидание на отрезке от 0 до 1 равно 0.5

Дисперсия определяется по формуле(рис. 2):

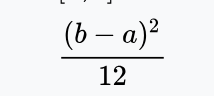


Рисунок 2 — Формула дисперсии

Дисперсия на отрезке от 0 до 1 равна 0.08(3)

**Вывод**

В данной работе было получено приблизительное значение числа π. Также было установлена зависимость точности вычисления числа π от размера выборки, то есть чем больше выборка, тем точнее будет оценка числа π. Помимо этого, установлено что плотность распределения случайных величин у выбранного генератора случайных чисел близка к равномерному распределению.