МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе №3**

**«**Программирование циклического вычислительного процесса**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-324», «АВТФ» *Ассистент кафедры ЗИ*

*Клаус В. А. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2024 г.«\_\_\_» \_\_\_\_\_\_ 2024 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы**: изучение циклических алгоритмов, операторов цикла, программирование циклического вычислительного процесса.

**Задание к работе**: Реализовать циклический вычислительный процесс. Самостоятельно решить задачи в соответствии с индивидуальным вариантом.

**Задания:**

**Задание 1.** Вычислить и вывести на экран или в файл в виде таблицы значения функции, заданной графически, на интервале от Xнач до Xкон с шагом dx. Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблица должна иметь заголовком и шапку.

**Задание 2.** Тесты на простоту (реализовать метод перебора делителей, тесты Миллера, Поклингтона, ГОСТ Р 34.10-94

**Задание 3.** Решить задачу об остывании чашки кофе из кофейни НГТУ. Написать программу, моделирующую процесс остывания кофе. Программа должна позволять задавать все необходимые параметры. Построить графики или таблицы зависимостей величин. Считаем, что температуру кофе измеряли через некоторые, необязательно равномерные промежутки времени. Написать программу, которая бы по данным измерений строила линейную модель, вычисляя статистические критерии их адекватности.

**Задание 1**

**C++:**

**#include <iostream>**

**#include <math.h>**

**using namespace std;**

**void pudge(float start, float end, float step) { //**

**for (float x = start; x <= end; x += step) {**

**float result;**

**if (x >= -10 && x <= -6) {**

**result = round((-1\*(sqrt(4-(pow(x+8, 2))))+2)\*100) / 100;**

**cout << x << "\t" << result << endl;**

**}**

**if (x > -6 && x <= -4) {**

**result = 2;**

**cout << x << "\t" << result << endl;**

**}**

**if (x > -4 && x <= 2) {**

**result = -0.5 \* x;**

**if (result == -0) result = abs(result);**

**cout << x << "\t" << result << endl;**

**}**

**if (x > 2) {**

**result = x - 3;**

**cout << x << "\t" << result << endl;**

**}**

**}**

**}**

**int main(){**

**setlocale(LC\_ALL, "Rus");**

**float xstart, xend, step;**

**cout << "Введите начальный x: ";**

**cin >> xstart;**

**cout << "Введите конечный x: ";**

**cin >> xend;**

**cout << "Введите шаг: ";**

**cin >> step;**

**cout << "X\t" << "Y\n";**

**pudge(xstart, xend, step);**

**return 0;**

**}**

**Input: Введите начальный x: -10**

**Введите конечный x: 4**

**Введите шаг: 2**

**X Y**

**-10 2**

**-8 0**

**-6 2**

**-4 2**

**-2 1**

**0 0**

**2 -1**

**4 1**

**Python:**

**import numpy as np**

**import math**

**def pudge(start, end, step):**

**for x in np.arange(start, end+step, step):**

**if -10 <= x <= -6:**

**result = round((-1\*(math.sqrt(4-(math.pow(x+8, 2))))+2)\*100) / 100**

**print(x, "\t", result)**

**elif -6 < x <= -4:**

**result = 2**

**print(x, "\t", result)**

**elif -4 < x <= 2:**

**result = -0.5 \* x**

**if result == -0:**

**result = abs(result)**

**print(x, "\t", result)**

**elif x > 2:**

**result = x - 3**

**print(x, "\t", result)**

**xstart = int(input("Введите начальный x: "))**

**xend = int(input("Введите конечный x: "))**

**step = float(input("Введите шаг: "))**

**print("X\tY")**

**pudge(xstart, xend, step)**

**Input: Введите начальный x: -10**

**Введите конечный x: 4**

**Введите шаг: 2**

**X Y**

**-10 2**

**-8 0**

**-6 2**

**-4 2**

**-2 1**

**0 0**

**2 -1**

**4 1**

**Задание 2**

**С++:**

**#include <iostream>**

**#include <math.h>**

**#include <vector>**

**#include <iomanip>**

**#include <cstdint>**

**using namespace std;**

**vector<int> Eratos(int num) { // решето Эратосфена**

**vector<int> numbersToNum;**

**vector<int> Prostie;**

**for (int i = 0; i < num; i++) {**

**numbersToNum.push\_back(i);**

**}**

**for (int i = 2; i < num; i++) {**

**if (numbersToNum[i] != 0) {**

**for (int j = i + numbersToNum[i]; j < num; j += numbersToNum[i]) {**

**numbersToNum[j] = 0;**

**}**

**}**

**}**

**for (int i = 0; i < num; i++) {**

**if (numbersToNum[i] != 0 && numbersToNum[i] != 1) {**

**Prostie.push\_back(numbersToNum[i]);**

**}**

**}**

**return Prostie;**

**}**

**int Moded(int number, int m) {**

**int result = 1;**

**for (int i = 0; i < m; i++) {**

**result \*= number;**

**result %= (m + 1);**

**}**

**return result;**

**}**

**int Generator(vector<int> numberProst, int n, vector<int>& number, vector<int>& dividers, int R) { // Генератор**

**int m = (n - 1) / R;**

**int i = 0;**

**while (m > 1) {**

**if (m % numberProst[i] == 0) {**

**dividers.push\_back(numberProst[i]);**

**m /= numberProst[i];**

**}**

**else {**

**i++;**

**}**

**}**

**int t = 5;**

**for (int j = 0; j < t; j++) {**

**number.push\_back(rand() % (n));**

**}**

**for (int j : number) {**

**if ((Moded(j, n - 1) % n) != 1) {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**int Miller(vector<int> numberProst, int n) { // Тест Миллера**

**vector<int> dividers;**

**vector<int> number;**

**int test = Generator(numberProst, n, number, dividers, 2);**

**if (test == 0) return 0;**

**int k = 0;**

**for (int j : dividers) {**

**for (int z : number) {**

**if ((Moded(z, (n - 1) / j) % n) != 1) {**

**k++;**

**break;**

**}**

**}**

**}**

**if (k == 0) {**

**return 0;**

**}**

**return 1;**

**}**

**int Poli(vector<int> numberProst, int n) { // Тест Поклингтона**

**int k = 0;**

**vector<int> dividers;**

**vector<int> number;**

**int R = 4;**

**if ((n / 8) != 0) {**

**R = rand() % (n / 8) \* 2;**

**if (R == 0) {**

**R = 4;**

**}**

**}**

**int test = Generator(numberProst, n, number, dividers, R);**

**if (test == 0)return 0;**

**for (int z : number) {**

**for (int j : dividers) {**

**if ((Moded(z, (n - 1) / j) % n) == 1) {**

**k++;**

**break;**

**}**

**}**

**}**

**if (k == 0) {**

**return 1;**

**}**

**return 0;**

**}**

**int64\_t power(int a, int b) { // степень**

**int64\_t s = 1;**

**for (int i = 0; i < b; i++) {**

**s = s \* a;**

**}**

**return s;**

**}**

**int powerMod(int a, int b, int m) { // a^x mod p**

**int s = 1;**

**for (int i = 0; i < b; i++) {**

**s = s \* a;**

**s = s % m;**

**}**

**return s;**

**}**

**int GOST(int t, int q) { // ГОСТ Р 34.10 - 94.**

**bool f = false;**

**int p = 0;**

**while (true) {**

**int N = power(2, t - 1) / q;**

**if (N % 2 == 1)N++;**

**int u = 0;**

**while (true) {**

**p = (N + u) \* q + 1;**

**if (power(2, t) < p) {**

**break;**

**}**

**if ((power(2, p - 1) % p == 1) && (power(2, N + u) % p != 1)) {**

**f = true;**

**break;**

**}**

**u = u + 2;**

**}**

**if (f) {**

**return p;**

**}**

**}**

**}**

**int MillerRabin(int number, int k) { // Миллер-Рабин**

**if (number == 2 || number == 3) return 1;**

**if (number < 2 || number % 2 == 0) return 0;**

**int d = number - 1;**

**int s = 0;**

**int y = 0;**

**while (d % 2 == 0)**

**{**

**d /= 2;**

**s += 1;**

**}**

**for (int i = 0; i < k; i++) {**

**int a = rand() % (number - 4) + 2;**

**int x = powerMod(a, d, number);**

**for (int j = 0; j < s; j++) {**

**y = (x \* x) % number;**

**if (y == 1 && x != 1 && x != (number - 1)) {**

**return 0;**

**}**

**x = y;**

**}**

**if (y != 1) {**

**return 0;**

**}**

**}**

**return 1;**

**}**

**int main() {**

**setlocale(LC\_ALL, "Rus");**

**srand(time(0));**

**int num = 500;**

**vector<int> Prostie = Eratos(num);**

**cout << "Простые числа до " << num << ": " << endl;**

**for (int i : Prostie) {**

**cout << setw(4) << i;**

**}**

**cout << endl;**

**int t = 5;**

**int q = 7;**

**if (q < power(2, t / 2 + 1)) {**

**int number = GOST(t, q);**

**cout << number << endl;**

**}**

**vector<int> ReallyProstie;**

**vector<int> BadOnes;**

**int MissedOnes = 0;**

**int TotalNums = 0;**

**while (ReallyProstie.size() < 10) {**

**int test = rand() % (500 - 4) + 2;**

**if (Miller(Prostie, test) && Poli(Prostie, test)) {**

**ReallyProstie.push\_back(test);**

**BadOnes.push\_back(MissedOnes);**

**MissedOnes = 0;**

**}**

**else {**

**MissedOnes += MillerRabin(test, 10);**

**}**

**TotalNums++;**

**}**

**cout << "Счетчик простых отвергнутых чисел: ";**

**for (int i : BadOnes) {**

**cout << i << " ";**

**}**

**MissedOnes = 0;**

**cout << endl;**

**cout << "Пройдено чисел: " << TotalNums << endl;**

**cout << "10 Получившихся чисел:" << endl;**

**for (int i : ReallyProstie) {**

**cout << setw(5) << i;**

**MissedOnes = MillerRabin(i, 10);**

**if (MissedOnes == 1) {**

**cout << " +";**

**}**

**else {**

**cout << " -";**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Input: Простые числа до 500:**

**2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499**

**29**

**Счетчик простых отвергнутых чисел: 0 3 4 0 0 2 2 2 5 8**

**Пройдено чисел: 189**

**10 Получившихся чисел:**

**89 +**

**107 +**

**191 +**

**79 +**

**461 +**

**367 +**

**281 +**

**367 +**

**347 +**

**163 +**

**Python:**

**import random**

**import math**

**def Eratos(num): # решето Эратосфена**

**numbersToNum = [i for i in range(num)]**

**Prostie = []**

**for i in range(2, num):**

**if numbersToNum[i] != 0:**

**for j in range(i + numbersToNum[i], num, numbersToNum[i]):**

**numbersToNum[j] = 0**

**for i in range(num):**

**if numbersToNum[i] != 0 and numbersToNum[i] != 1:**

**Prostie.append(numbersToNum[i])**

**return Prostie**

**def Moded(number, m):**

**result = 1**

**for i in range(m):**

**result \*= number**

**result %= (m + 1)**

**return result**

**def Generator(numberProst, n, number, dividers, R): # Генератор**

**m = (n - 1) // R**

**i = 0**

**while m > 1:**

**if m % numberProst[i] == 0:**

**dividers.append(numberProst[i])**

**m //= numberProst[i]**

**else:**

**i += 1**

**t = 5**

**for j in range(t):**

**number.append(random.random() % n)**

**for j in number:**

**if (Moded(j, n - 1) % n) != 1:**

**return 0**

**return 1**

**def Miller(numberProst, n): # тест Миллера**

**dividers = []**

**number = []**

**test = Generator(numberProst, n, number, dividers, 2)**

**if test == 0:**

**return 0**

**k = 0**

**for j in dividers:**

**for z in number:**

**if (Moded(z, (n - 1) // j) % n) != 1:**

**k += 1**

**break**

**if k == 0:**

**return 0**

**return 1**

**def Poli(numberProst, n): # тест Поклингтона**

**k = 0**

**dividers = []**

**number = []**

**R = 4**

**if (n // 8) != 0:**

**R = random.randint(0, n // 8) \* 2**

**if R == 0:**

**R = 4**

**test = Generator(numberProst, n, number, dividers, R)**

**if test == 0:**

**return 0**

**for z in number:**

**for j in dividers:**

**if (Moded(z, (n - 1) // j) % n) == 1:**

**k += 1**

**break**

**if k == 0:**

**return 1**

**return 0**

**def power(a, b): # степень**

**s = 1**

**for i in range(b):**

**s = s \* a**

**return s**

**def powerMod(a, b, m): # a^x mod p**

**s = 1**

**for i in range(b):**

**s = s \* a**

**s = s % m**

**return s**

**def GOST(t, q): # ГОСТ Р 34.10 - 94.**

**f = False**

**p = 0**

**while True:**

**N = power(2, t - 1) // q**

**if N % 2 == 1:**

**N += 1**

**u = 0**

**while True:**

**p = (N + u) \* q + 1**

**if power(2, t) < p:**

**break**

**if (power(2, p - 1) % p == 1) and (power(2, N + u) % p != 1):**

**f = True**

**break**

**u = u + 2**

**if f:**

**return p**

**def MillerRabin(number, k): # Миллер-Рабин**

**if number == 2 or number == 3:**

**return 1**

**if number < 2 or number % 2 == 0:**

**return 0**

**d = number - 1**

**s = 0**

**y = 0**

**while d % 2 == 0:**

**d //= 2**

**s += 1**

**for i in range(k):**

**a = random.randint(2, number-4)**

**x = powerMod(a, d, number)**

**for j in range(s):**

**y = (x \* x) % number**

**if y == 1 and x != 1 and x != (number - 1):**

**return 0**

**x = y**

**if y != 1:**

**return 0**

**return 1**

**num = 500**

**Prostie = Eratos(num)**

**print("Простые числа до", num, ": ")**

**for i in Prostie:**

**print("{:4}".format(i), end="")**

**print()**

**t = 5**

**q = 7**

**if q < power(2, t // 2 + 1):**

**number = GOST(t, q)**

**print(number)**

**ReallyProstie = []**

**BadOnes = []**

**MissedOnes = 0**

**TotalNums = 0**

**while len(ReallyProstie) < 10:**

**test = random.randint(2, 500-4)**

**if Miller(Prostie, test) and Poli(Prostie, test):**

**ReallyProstie.append(test)**

**BadOnes.append(MissedOnes)**

**MissedOnes = 0**

**else:**

**MissedOnes += MillerRabin(test, 10)**

**TotalNums += 1**

**print("Счетчик простых отвергнутых чисел:", end=" ")**

**for i in BadOnes:**

**print(i, end=" ")**

**MissedOnes = 0**

**print()**

**print("Пройдено чисел:", TotalNums)**

**print("10 Получившихся чисел:")**

**for i in ReallyProstie:**

**print("{:5}".format(i), end="")**

**MissedOnes = MillerRabin(i, 10)**

**if MissedOnes == 1:**

**print(" +")**

**else:**

**print(" -")**

**Input: Простые числа до 500:**

**2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499**

**29**

**Счетчик простых отвергнутых чисел: 0 3 4 0 0 2 2 2 5 8**

**Пройдено чисел: 189**

**10 Получившихся чисел:**

**89 +**

**107 +**

**191 +**

**79 +**

**461 +**

**367 +**

**281 +**

**367 +**

**347 +**

**163 +**

**Задание 3**

**C++:**

**#include <iostream>**

**#include <vector>**

**#include <iomanip>**

**#include <numeric>**

**#include <cmath>**

**using namespace std;**

**vector<double> coffee(double Tk, double Tsr, double r, int time) { // высчитывает температуру кофе со временем**

**vector<double> temperatures; // вектор для температуры**

**for (int i = 1; i <= time; i++) {**

**double t = Tsr + (Tk - Tsr) \* exp(-r \* i); // вычисление по закону теплопроводности Ньютона**

**temperatures.push\_back(t);**

**}**

**return temperatures;**

**}**

**pair<double, double> aprox(const vector<double>& x, const vector<double>& y) { // значение аппроксимирующей прямой**

**double sumx = accumulate(x.begin(), x.end(), 0.0); // сумма всех х**

**double sumy = accumulate(y.begin(), y.end(), 0.0); // сумма всех у**

**double xy = inner\_product(x.begin(), x.end(), y.begin(), 0.0); // сумма произведения всех х на у**

**double xx = inner\_product(x.begin(), x.end(), x.begin(), 0.0); // сумма всех х в квадрате**

**double n = x.size();**

**pair <double, double> approximatingLine; // пара значений аппроксимирующей прямой**

**approximatingLine.first = (n \* xy - sumx \* sumy) / (n \* xx - sumx \* sumx); // значение а**

**approximatingLine.second = (sumy - approximatingLine.first \* sumx) / n; // значение b**

**return approximatingLine;**

**}**

**double korrel(const vector<double>& x, const vector<double>& y) { // коэффициент корреляции**

**double xsr = accumulate(x.begin(), x.end(), 0.0) / x.size(); // среднее значение х**

**double ysr = accumulate(y.begin(), y.end(), 0.0) / y.size(); // среднее значение у**

**double sumxy = 0, sumxx = 0, sumyy = 0;**

**for (int i = 0; i < x.size(); i++) {**

**sumxy += (x[i] - xsr) \* (y[i] - ysr); // сумма произведения разности всех х и х среднего на разность всех у на у среднего**

**sumxx += (x[i] - xsr) \* (x[i] - xsr); // сумма квадрата разности всех х и х среднего**

**sumyy += (y[i] - ysr) \* (y[i] - ysr); // сумма квадрата разности всех у и у среднего**

**}**

**return sumxy / sqrt(sumxx \* sumyy); // подсчёт коэффициента**

**}**

**int main() {**

**setlocale(LC\_ALL, "Russian");**

**double Tk = 90; // температура кофе**

**double Tsr = 25; // температура окружающей среды**

**double r = 0.005; // коэффициент остывания**

**int time = 60; // время остывания в минутах**

**vector<double> temperatures = coffee(Tk, Tsr, r, time); // заполнение вектора с температурой**

**vector<double> times; // вектор для времени**

**cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;**

**cout << "|" << setw(3) << "time (x)|" << setw(5) << "temperature (y)|" << endl; // шапка таблицы**

**for (int i = 0, j = 1; i < temperatures.size() && j <= time; i++, j++) {**

**cout << "|" << setw(5) << j << setw(4) << "|";**

**cout << setw(9) << temperatures[i] << setw(7) << "|" << endl; // вывод значений в табличной форме**

**times.push\_back(i); // заполнение вектора со временем**

**}**

**cout << "Аппроксимирующая прямая: a = ";**

**pair<double, double> approximatingLine = aprox(times, temperatures); // заполнение пары со значением аппроксимирующей прямой**

**cout << approximatingLine.first << " b = " << approximatingLine.second << endl;**

**cout << "Коэффициент корреляции: r = " << korrel(times, temperatures);**

**return 0;**

**}**

**Input:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**|time (x)|temperature (y)|**

**| 1 | 89.6758 |**

**| 2 | 89.3532 |**

**| 3 | 89.0323 |**

**| 4 | 88.7129 |**

**| 5 | 88.3951 |**

**| 6 | 88.079 |**

**| 7 | 87.7644 |**

**| 8 | 87.4513 |**

**| 9 | 87.1398 |**

**| 10 | 86.8299 |**

**| 11 | 86.5215 |**

**| 12 | 86.2147 |**

**| 13 | 85.9094 |**

**| 14 | 85.6056 |**

**| 15 | 85.3033 |**

**| 16 | 85.0026 |**

**| 17 | 84.7033 |**

**| 18 | 84.4055 |**

**| 19 | 84.1092 |**

**| 20 | 83.8144 |**

**| 21 | 83.5211 |**

**| 22 | 83.2292 |**

**| 23 | 82.9388 |**

**| 24 | 82.6498 |**

**| 25 | 82.3623 |**

**| 26 | 82.0762 |**

**| 27 | 81.7915 |**

**| 28 | 81.5083 |**

**| 29 | 81.2264 |**

**| 30 | 80.946 |**

**| 31 | 80.667 |**

**| 32 | 80.3893 |**

**| 33 | 80.1131 |**

**| 34 | 79.8382 |**

**| 35 | 79.5647 |**

**| 36 | 79.2926 |**

**| 37 | 79.0218 |**

**| 38 | 78.7523 |**

**| 39 | 78.4843 |**

**| 40 | 78.2175 |**

**| 41 | 77.9521 |**

**| 42 | 77.688 |**

**| 43 | 77.4252 |**

**| 44 | 77.1637 |**

**| 45 | 76.9036 |**

**| 46 | 76.6447 |**

**| 47 | 76.3871 |**

**| 48 | 76.1308 |**

**| 49 | 75.8758 |**

**| 50 | 75.6221 |**

**| 51 | 75.3696 |**

**| 52 | 75.1184 |**

**| 53 | 74.8684 |**

**| 54 | 74.6197 |**

**| 55 | 74.3722 |**

**| 56 | 74.1259 |**

**| 57 | 73.8809 |**

**| 58 | 73.6371 |**

**| 59 | 73.3946 |**

**| 60 | 73.1532 |**

**Аппроксимирующая прямая: a = -0.27966 b = 89.2657**

**Коэффициент корреляции: r = -0.999252**

**Python:**

**import math**

**from typing import List, Tuple**

**def coffee(Tk: float, Tsr: float, r: float, time: int) -> List[float]: # высчитывает температуру кофе со временем**

**temperatures = [] # вектор для температуры**

**for i in range(1, time+1):**

**t = Tsr + (Tk - Tsr) \* math.exp(-r \* i) # вычисление по закону теплопроводности Ньютона**

**temperatures.append(t)**

**return temperatures**

**def aprox(x: List[float], y: List[float]) -> Tuple[float, float]: # значение аппроксимирующей прямой**

**sumx = sum(x) # сумма всех х**

**sumy = sum(y) # сумма всех у**

**xy = sum([a\*b for a, b in zip(x, y)]) # сумма произведения всех х на у**

**xx = sum([a\*\*2 for a in x]) # сумма всех х в квадрате**

**n = len(x)**

**a = (n \* xy - sumx \* sumy) / (n \* xx - sumx\*\*2) # пара значений аппроксимирующей прямой**

**b = (sumy - a \* sumx) / n**

**return a, b**

**def korrel(x: List[float], y: List[float]) -> float: # коэффициент корреляции**

**xsr = sum(x) / len(x) # среднее значение х**

**ysr = sum(y) / len(y) # среднее значение у**

**sumxy = sum([(a - xsr) \* (b - ysr) for a, b in zip(x, y)]) # сумма произведения разности всех х и х среднего на разность всех у на у среднего**

**sumxx = sum([(a - xsr)\*\*2 for a in x]) # сумма квадрата разности всех х и х среднего**

**sumyy = sum([(b - ysr)\*\*2 for b in y]) # сумма квадрата разности всех у и у среднего**

**return sumxy / math.sqrt(sumxx \* sumyy) # подсчёт коэффициента**

**Tk = 90 # температура кофе**

**Tsr = 25 # температура окружающей среды**

**r = 0.005 # коэффициент остывания**

**time = 60 # время остывания в минутах**

**temperatures = coffee(Tk, Tsr, r, time) # заполнение вектора с температурой**

**times = [] # вектор для времени**

**print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")**

**print("|" + "{:^8}".format("time (x)") + "|" + "{:^16}".format("temperature (y)") + "|") # шапка таблицы**

**for i in range(len(temperatures)):**

**j = i + 1**

**print("|" + "{:^8}".format(str(j)) + "|" + "{:^16}".format(str(temperatures[i])) + "|") # вывод значений в табличной форме**

**times.append(i) # заполнение вектора со временем**

**print("Аппроксимирующая прямая: a = ", end="")**

**approximatingLine = aprox(times, temperatures) # заполнение пары со значением аппроксимирующей прямой**

**print(approximatingLine[0], " b = ", approximatingLine[1])**

**print("Коэффициент корреляции: r = ", korrel(times, temperatures))**

**Input:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**|time (x)|temperature (y)|**

**| 1 | 89.6758 |**

**| 2 | 89.3532 |**

**| 3 | 89.0323 |**

**| 4 | 88.7129 |**

**| 5 | 88.3951 |**

**| 6 | 88.079 |**

**| 7 | 87.7644 |**

**| 8 | 87.4513 |**

**| 9 | 87.1398 |**

**| 10 | 86.8299 |**

**| 11 | 86.5215 |**

**| 12 | 86.2147 |**

**| 13 | 85.9094 |**

**| 14 | 85.6056 |**

**| 15 | 85.3033 |**

**| 16 | 85.0026 |**

**| 17 | 84.7033 |**

**| 18 | 84.4055 |**

**| 19 | 84.1092 |**

**| 20 | 83.8144 |**

**| 21 | 83.5211 |**

**| 22 | 83.2292 |**

**| 23 | 82.9388 |**

**| 24 | 82.6498 |**

**| 25 | 82.3623 |**

**| 26 | 82.0762 |**

**| 27 | 81.7915 |**

**| 28 | 81.5083 |**

**| 29 | 81.2264 |**

**| 30 | 80.946 |**

**| 31 | 80.667 |**

**| 32 | 80.3893 |**

**| 33 | 80.1131 |**

**| 34 | 79.8382 |**

**| 35 | 79.5647 |**

**| 36 | 79.2926 |**

**| 37 | 79.0218 |**

**| 38 | 78.7523 |**

**| 39 | 78.4843 |**

**| 40 | 78.2175 |**

**| 41 | 77.9521 |**

**| 42 | 77.688 |**

**| 43 | 77.4252 |**

**| 44 | 77.1637 |**

**| 45 | 76.9036 |**

**| 46 | 76.6447 |**

**| 47 | 76.3871 |**

**| 48 | 76.1308 |**

**| 49 | 75.8758 |**

**| 50 | 75.6221 |**

**| 51 | 75.3696 |**

**| 52 | 75.1184 |**

**| 53 | 74.8684 |**

**| 54 | 74.6197 |**

**| 55 | 74.3722 |**

**| 56 | 74.1259 |**

**| 57 | 73.8809 |**

**| 58 | 73.6371 |**

**| 59 | 73.3946 |**

**| 60 | 73.1532 |**

**Аппроксимирующая прямая: a = -0.27966 b = 89.2657**

**Коэффициент корреляции: r = -0.999252**

**Вывод:**

В ходе выполнения лабораторной работы №3 я познакомился с новыми функциями, библиотеками языков C++ и Python. Повторил математические знания и реализовал их в коде. Познакомился с новыми тестами на простоту, которые пригодятся в дальнейшем изучении Криптографии.

**Ссылка на репозиторий:** [нажмите](https://github.com/JamesSevil/laba3)