МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**ЛАБОРАТОРНАЯ РАБОТА №4**

**«Последовательные одномерные контейнеры»**

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

Выполнил:Проверил:

Студент гр. «АБс-322», «АВТФ *Ассистент кафедры ЗИ*

*Батталов Игорь Павлович Исаев Г. А.*

«» 2024г«\_\_\_» \_\_\_\_\_\_ 2024 г.

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

(подпись) (подпись)

Новосибирск 2024

# Цель

Изучение алгоритмов формирования и обработки одномерных массивов и последовательных контейнеров, программирование и отладка программ формирования и обработки массивов.

# Задание

Написать программу решения задачи в соответствии с индивидуальным вариантом.

**Задание 1.** «Методы решения нелинейный уравнений»

**Задание 2.** «Вычисления в одномерных массивах».

1. Напишите программу, в которой определен массив 𝑎𝑟𝑟1 из 𝑛 чисел (15 ≤ 𝑛 ≤ 30) и инициализирован целыми случайными числами из диапазона [10, 30]. Определите массив 𝑎𝑟𝑟2 из 5 чисел и инициализируйте массив 𝑎𝑟𝑟2 целыми случайными числами из диапазона [10, 30].

2. Выведите количество чётных элементов массива 𝑎𝑟𝑟1 на нечётных местах, а также сами элементы. 15

3. Определите, какие числа из массива 𝑎𝑟𝑟2 встречаются в массиве 𝑎𝑟𝑟1. Выведите эти числа и сколько раз они встречаются в массиве 𝑎𝑟𝑟1. Если ни одно число не встречается, то выведите соответствующее сообщение.

4. Напишите программу, модифицирующую массив случайных символов размера 𝑛 , как в примере: [v,!,N,R,(,3] → [!,N,R,(,3,v].

5. Определите и инициализируйте массив, состоящий из случайных целых чисел, входящих в диапазон [1000, 9000]. Вычислите сумму элементов массива. Напишите программу, меняющую местами цифры в элементах массива в порядке возрастания (например, 8315 → 1358). Отсортируйте массив в порядке возрастания. Вычислите сумму элементов массива и сравните с суммой элементов исходного массива.

**Задание 3.** «Работа с генераторами рандомных чисел». Задание выполняется на языке C++. В соответствии с вариантом необходимо описать распределение генератора, с помощью этого генератора заполнить 3 массива размером 50, 100 и 1000 соответственно числами от 1 до 100.

Использовать критерий хи-квадрат для проверки гипотезы о нормальном распределении выборки.

Проанализировать значение критерия хи-квадрат и вывести: 1) результат проверки гипотезы; 2) ожидаемое и реальное математическое ожидание.

**Задание 4.** «Дилемма заключенного»\*. Требуется реализовать игру «Предать или сотрудничать» и реализовать 3 алгоритма поведения в игре.

**Задание 5.** «Генерация псевдослучайных чисел». Вариант 2 (Генератор псевдослучайных чисел на основе алгоритма BBS).

**Задание 1**

**С++**

# Листинг программы

#include <iostream>

#include<iomanip>

using namespace std;

float linefunc(float x) {//Функция считает Y первой прямой на графике

return 3.0;

}

float cirklfunc(float x) {//Функция считает Y полукруга

return (6.0 - sqrt(36.0 - 4.0 \* pow(x, 2.0))) / 2.0;

}

float linefunc2(float x) {//Функция считает Y второй прямой на графике

return -2.0 \* x + 9.0;

}

float linefunc3(float x) {//Функция считает Y третьей прямой на графике

return x - 9.0;

}

int main() {

cout << fixed << setprecision(2);

cout << "Number 1" << endl;

cout << "x y" << endl;

for (float x = -7.0; x <= 11.0; x = x + 0.5) {

if ((x >= -7.0) and (x <= -3.0)) {

cout << "x = " << x << " " << "y = " << linefunc(x) << endl;

}

if ((x > -3.0) and (x < 3.0)) {

cout << "x = " << x << " " << "y = " << cirklfunc(x) << endl;

}

if ((x >= 3.0) and (x <= 6.0)) {

cout << "x = " << x << " " << "y = " << linefunc2(x) << endl;

}

if ((x > 6.0) and (x <= 11.0)) {

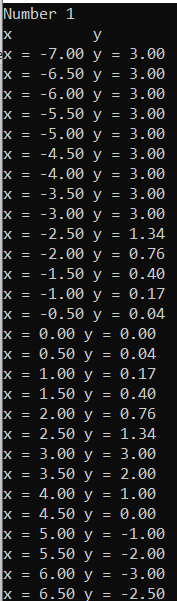
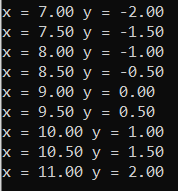
cout << "x = " << x << " " << "y = " << linefunc3(x) << endl;

}

}

}

# Результат работы программы



# С#

**Листинг программы**

using System;

using System.Text;

namespace CSharpTranslation

{

class Program

{

static float linefunc(float x)

{

return 3.0f;

}

static float cirklfunc(float x)

{

return (6.0f - (float)Math.Sqrt(36.0f - 4.0f \* (float)Math.Pow(x, 2.0f))) / 2.0f;

}

static float linefunc2(float x)

{

return -2.0f \* x + 9.0f;

}

static float linefunc3(float x)

{

return x - 9.0f;

}

static void Main(string[] args)

{

Console.OutputEncoding = Encoding.UTF8;

Console.WriteLine("Number 1");

Console.WriteLine("x y");

for (float x = -7.0f; x <= 11.0f; x += 0.5f)

{

if ((x >= -7.0f) && (x <= -3.0f))

{

Console.WriteLine($"x = {x:F2} y = {linefunc(x):F2}");

}

else if ((x > -3.0f) && (x < 3.0f))

{

Console.WriteLine($"x = {x:F2} y = {cirklfunc(x):F2}");

}

else if ((x >= 3.0f) && (x <= 6.0f))

{

Console.WriteLine($"x = {x:F2} y = {linefunc2(x):F2}");

}

else if ((x > 6.0f) && (x <= 11.0f))

{

Console.WriteLine($"x = {x:F2} y = {linefunc3(x):F2}");

}

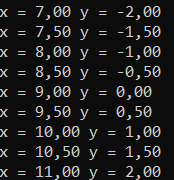
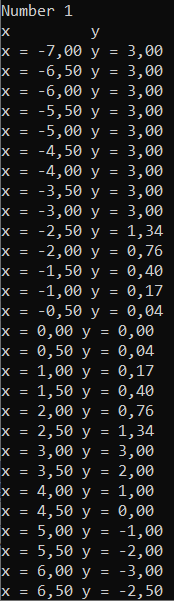
}

}

}

}

**Результаты работы программы**



**Задание 2**

**C++**

**Листинг программы**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <set>

#include <map>

using namespace std;

void resheto(){

int n;

cout << "Input amount of prime number ";

cin >> n;

vector<int> prostchisla;

for (int i = 0; i < n + 1; i++)

prostchisla.push\_back(i);

for (int p = 2; p < n + 1; p++)

{

if (prostchisla[p] != 0)

{

for (int j = p \* p; j < n + 1; j += p)

prostchisla[j] = 0;

}

}

int linecount = 0;

for (int al : prostchisla) {

if (al != 0) {

if (linecount == 10) {

cout << "\n";

linecount = 0;

}

cout << al << " ";

linecount += 1;

}

}

}

unsigned long long mod(unsigned long long number, unsigned long long power, unsigned long long n) {

unsigned long long res = 1;

while (power) {

if (power % 2) res = ( res \* number) % n;

number = ( number \* number) % n;

power /= 2;

}

return res;

}

void tableprost(vector<int>& pch) {//Функция заполнения вектора простыми числами из файла (Используем готовую таблицу)

ifstream file("chisla.txt");

if (file.is\_open()) {

while (true) {

string stroka;

getline(file, stroka, ' ');

if (stroka == "@") break;

pch.push\_back(stoi(stroka));

}

}

}

bool TestMillerasver(int t, int N, set<int> q, int& K) {//Тест Миллера на простоту

int countT = 0;

int countF = 0;

for (int j = 0; j < 50; j++) {

vector<int> aj;//Создание случайных чисел

for (int i = 0; i < t; i++) {

int rand1 = rand() % N;

if (rand1 == 0) {

rand1 = 2 + 1;

}

aj.push\_back(rand1);

}

int iscomposite = 0;

for (int aj1 : aj) {

if (mod(aj1, N - 1, N) != 1) {

iscomposite = 1;

break;

}

}

if (iscomposite == 1) {

countF = countF + 1;

continue;

}

for (int aj1 : aj) {

for (int qi : q) {

if (mod(aj1, (N - 1) / qi, N) != 1) {

iscomposite = 1;

continue;

}

}

}

if (iscomposite == 0) {

countT = countT + 1;

continue;

}

countT = countT + 1;

}

if (countT > countF) return true;

else {

K += 1;

return false;

}

}

bool TestPoklingtona(int t, int N, set<int> q, int& K, vector<int> prostch) {//Проверка чисел на простоту. Тест Поклингтона

int countT = 0;

int countF = 0;

for (int j = 0; j < 100; j++) {

vector<int> aj;//Создание случайных чисел

for (int i = 0; i < t; i++) {

int rand1 = rand() % N;

if (rand1 == 0) {

continue;

}

aj.push\_back(rand1);

}

int iscomposite = 0;

for (int aj1 : aj) {

if (mod(aj1, N - 1, N) != 1) {

iscomposite = 1;

break;

}

}

if (iscomposite == 1) {

countF += 1;

continue;

}

for (int aj1 : aj) {

for (int qj : q) {

if (mod(aj1, (N - 1) / qj, N) == 1) {

iscomposite = 1;

break;

}

}

if (iscomposite == 1) break;

}

if (iscomposite == 1) {

countF += 1;

continue;

}

if (iscomposite == 0) {

countT += 1;

}

}

if (countT > countF) {

return true;

}

K += 1;

return false;

}

void GenMillera(int size, vector<int> pch, vector<int>& pchM, vector<int>& KM) {//Процедура получения простых чисел на основе теста Миллера

set<int> similar;

int K = 0;

for (int countcikl = 0; countcikl != 10;) {

set<int> q;

for (int i = 0; i < 2; i++) {

q.insert(pch[rand() % 93]);

}

int m = 1;

for (int chislo : q) {

m = m \* chislo;

}

int n = 2 \* m + 1;

int n1 = n;

int countn1 = 0;

while (n1 > 0) {

countn1 += 1;

n1 = n1 / 10;

}

for (int sim : similar) {

if (n == sim) countn1 = -1;

}

if (countn1 != size) continue;

if (TestMillerasver(1, n, q, K)) {

pchM.push\_back(n);

similar.insert(n);

countcikl += 1;

KM.push\_back(K);

K = 0;

}

}

}

void GenPoklingtona(int size, vector<int> pch, vector<int>& pchP, vector<int>& KP) {//Процедура получения простых чисел на основе теста Поклингтона

set<int> similar;

int K = 0;

for (int countcikl = 0; countcikl != 10;) {

set<int> q;

for (int i = 0; i < 2; i++) {

q.insert(pch[rand() % 93]);

}

int f = 1;

for (int chislo : q) {

f = f \* chislo;

}

int r = rand() % ((3 \* f) / 4);

if (r % 2 != 0) r = r + 1;

int n = r \* f + 1;

int n1 = n;

int countn1 = 0;

while (n1 > 0) {

countn1 += 1;

n1 = n1 / 10;

}

for (int sim : similar) {

if (n == sim) countn1 = -1;

}

if (countn1 != size) continue;

bool a = TestPoklingtona(1, n, q, K, pch);

if (TestPoklingtona(1, n, q, K, pch) == true) {

KP.push\_back(K);

pchP.push\_back(n);

similar.insert(n);

countcikl += 1;

K = 0;

}

}

}

void GenGOST(int size, vector<int> pch, vector<int>& pchG, vector<int>& KG) {//Процедура получения простых чисел

set<int> similar;

int K = 0;

for (int countcikl = 0; countcikl != 10;) {

int q = pch[rand() % 93];

K = 0;

for (int u = 0; true; u = u + 2) {

int N = ceil(pow(2, size - 1) / q);

if (N % 2 != 0) N = N + 1;

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

if (mod(2, p - 1, p) == 1 and mod(2, N + u, p) != 1) {

int n1 = p;

int countn1 = 0;

while (n1 > 0) {

countn1 += 1;

n1 = n1 / 10;

}

for (int sim : similar) {

if (p == sim) countn1 = -1;

}

if (countn1 == size) {

KG.push\_back(K);

pchG.push\_back(p);

countcikl += 1;

similar.insert(p);

}

break;

}

K += 1;

}

}

}

int main() {

int size = 3;//Размер простых чисел

vector<int> prostch;//Вектор простых чисел до 500

vector<int> pchM;

vector<int> pchP;

vector<int> pchG;

vector<int> KM;

vector<int> KP;

vector<int> KG;

tableprost(prostch);//Заполняем вектор простых чисел

cout << "Test Milera" << endl;

GenMillera(size, prostch, pchM, KM);

cout << "Num ";

for (int i = 1; i <= 10; i++) cout << i << " ";

cout << "\n";

cout << " p ";

for (int prch : pchM) {

cout << prch << " ";

}

cout << "\n";

for (int i = 0; i < 10; i++) {

if (i == 0) cout << " +";

else cout << " +";

}

cout << "\n";

cout << " K ";

for (int K : KM) {

cout << K << " ";

}

cout << "\n" << "Test Poklingtona" << endl;

GenPoklingtona(size, prostch, pchP, KP);

cout << "Num ";

for (int i = 1; i <= 10; i++) cout << i << " ";

cout << "\n";

cout << " p ";

for (int prch : pchP) {

cout << prch << " ";

}

cout << "\n";

for (int i = 0; i < 10; i++) {

if (i == 0) cout << " +";

else cout << " +";

}

cout << "\n";

cout << " K ";

for (int K : KP) {

cout << K << " ";

}

cout << "\n" << "GOST" << endl;

GenGOST(size, prostch, pchG, KG);

cout << "Num ";

for (int i = 1; i <= 10; i++) cout << i << " ";

cout << "\n";

cout << " p ";

for (int prch : pchG) {

cout << prch << " ";

}

cout << "\n";

for (int i = 0; i < 10; i++) {

if (i == 0) cout << " +";

else cout << " +";

}

cout << "\n";

cout << " K ";

for (int K : KG) {

cout << K << " ";

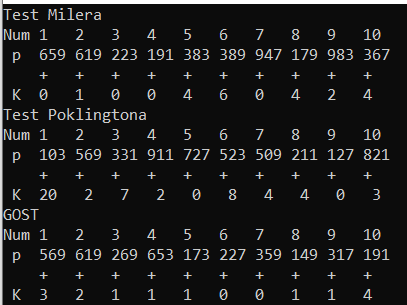
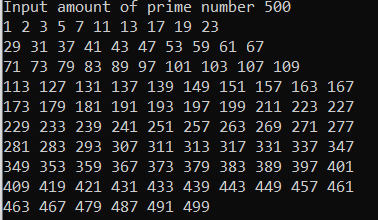
}

cout << "\n";

resheto();

}

**Результат работы программы**

**** ****

Решето Эратосфена

**С#**

**Листинг программы**

**using System;**

**using System.Collections.Generic;**

**using System.IO;**

**class Program**

**{**

**static void Resheto()**

**{**

**int n;**

**Console.WriteLine("Input amount of prime number ");**

**n = Convert.ToInt32(Console.ReadLine());**

**List<int> prostchisla = new List<int>();**

**for (int i = 0; i < n + 1; i++)**

**prostchisla.Add(i);**

**for (int p = 2; p < n + 1; p++)**

**{**

**if (prostchisla[p] != 0)**

**{**

**for (int j = p \* p; j < n + 1; j += p)**

**prostchisla[j] = 0;**

**}**

**}**

**int linecount = 0;**

**foreach (int al in prostchisla)**

**{**

**if (al != 0)**

**{**

**if (linecount == 10)**

**{**

**Console.WriteLine();**

**linecount = 0;**

**}**

**Console.Write(al + " ");**

**linecount += 1;**

**}**

**}**

**}**

**static ulong Mod(ulong number, ulong power, ulong n)**

**{**

**ulong res = 1;**

**while (power > 0)**

**{**

**if (power % 2 == 1)**

**res = (res \* number) % n;**

**number = (number \* number) % n;**

**power /= 2;**

**}**

**return res;**

**}**

**static bool TestMillerRabin(int t, int N, HashSet<int> q, ref int K)**

**{**

**int countT = 0;**

**int countF = 0;**

**Random rand = new Random();**

**for (int j = 0; j < 50; j++)**

**{**

**List<int> aj = new List<int>();**

**for (int i = 0; i < t; i++)**

**{**

**int rand1 = rand.Next(N);**

**if (rand1 == 0)**

**{**

**rand1 = 2 + 1;**

**}**

**aj.Add(rand1);**

**}**

**bool isComposite = false;**

**foreach (int aj1 in aj)**

**{**

**if (Mod((ulong)aj1, (ulong)(N - 1), (ulong)N) != 1)**

**{**

**isComposite = true;**

**break;**

**}**

**}**

**if (isComposite)**

**{**

**countF++;**

**continue;**

**}**

**foreach (int aj1 in aj)**

**{**

**foreach (int qi in q)**

**{**

**if (Mod((ulong)aj1, (ulong)((N - 1) / qi), (ulong)N) != 1)**

**{**

**isComposite = true;**

**break;**

**}**

**}**

**}**

**if (isComposite)**

**{**

**countF++;**

**continue;**

**}**

**countT++;**

**}**

**if (countT > countF)**

**return true;**

**else**

**{**

**K++;**

**return false;**

**}**

**}**

**static bool TestPocklington(int t, int N, HashSet<int> q, ref int K, List<int> prostch)**

**{**

**int countT = 0;**

**int countF = 0;**

**Random rand = new Random();**

**for (int j = 0; j < 100; j++)**

**{**

**List<int> aj = new List<int>();**

**for (int i = 0; i < t; i++)**

**{**

**int rand1 = rand.Next(N);**

**if (rand1 == 0)**

**{**

**continue;**

**}**

**aj.Add(rand1);**

**}**

**bool isComposite = false;**

**foreach (int aj1 in aj)**

**{**

**if (Mod((ulong)aj1, (ulong)(N - 1), (ulong)N) != 1)**

**{**

**isComposite = true;**

**break;**

**}**

**}**

**if (isComposite)**

**{**

**countF++;**

**}**

**foreach (int aj1 in aj)**

**{**

**foreach (int qj in q)**

**{**

**if (Mod((ulong)aj1, (ulong)((N - 1) / qj), (ulong)N) == 1)**

**{**

**isComposite = true;**

**break;**

**}**

**}**

**if (isComposite)**

**break;**

**}**

**if (isComposite)**

**{**

**countF++;**

**}**

**if (!isComposite)**

**{**

**countT++;**

**}**

**}**

**if (countT > countF)**

**{**

**return true;**

**}**

**K++;**

**return false;**

**}**

**static void GenMiller(int size, List<int> pch, ref List<int> pchM, ref List<int> KM)**

**{**

**HashSet<int> similar = new HashSet<int>();**

**int K = 0;**

**int countcikl = 0;**

**while (countcikl != 10)**

**{**

**HashSet<int> q = new HashSet<int>();**

**for (int i = 0; i < 2; i++)**

**{**

**q.Add(pch[new Random().Next(93)]);**

**}**

**int m = 1;**

**foreach (int chislo in q)**

**{**

**m \*= chislo;**

**}**

**int n = 2 \* m + 1;**

**int n1 = n;**

**int countn1 = 0;**

**while (n1 > 0)**

**{**

**countn1++;**

**n1 /= 10;**

**}**

**if (similar.Contains(n))**

**{**

**countn1 = -1;**

**}**

**if (countn1 != size)**

**{**

**continue;**

**}**

**if (TestMillerRabin(1, n, q, ref K))**

**{**

**pchM.Add(n);**

**similar.Add(n);**

**countcikl++;**

**KM.Add(K);**

**K = 0;**

**}**

**}**

**}**

**static void GenPocklington(int size, List<int> pch, ref List<int> pchP, ref List<int> KP)**

**{**

**HashSet<int> similar = new HashSet<int>();**

**int K = 0;**

**int countcikl = 0;**

**while (countcikl != 10)**

**{**

**HashSet<int> q = new HashSet<int>();**

**for (int i = 0; i < 2; i++)**

**{**

**q.Add(pch[new Random().Next(93)]);**

**}**

**int f = 1;**

**foreach (int chislo in q)**

**{**

**f \*= chislo;**

**}**

**int r = new Random().Next((3 \* f) / 4);**

**if (r % 2 != 0)**

**{**

**r++;**

**}**

**int n = r \* f + 1;**

**int n1 = n;**

**int countn1 = 0;**

**while (n1 > 0)**

**{**

**countn1++;**

**n1 /= 10;**

**}**

**if (similar.Contains(n))**

**{**

**countn1 = -1;**

**}**

**if (countn1 != size)**

**{**

**continue;**

**}**

**if (TestPocklington(1, n, q, ref K, pch))**

**{**

**KP.Add(K);**

**pchP.Add(n);**

**similar.Add(n);**

**countcikl++;**

**K = 0;**

**}**

**}**

**}**

**static void GenGOST(int size, List<int> pch, ref List<int> pchG, ref List<int> KG)**

**{**

**HashSet<int> similar = new HashSet<int>();**

**int K = 0;**

**int countcikl = 0;**

**while (countcikl != 10)**

**{**

**int q = pch[new Random().Next(93)];**

**K = 0;**

**for (int u = 0; ; u += 2)**

**{**

**int N = (int)Math.Ceiling(Math.Pow(2, size - 1) / q);**

**if (N % 2 != 0)**

**{**

**N++;**

**}**

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

**if (Mod(2, (ulong)(p - 1), (ulong)p) == 1 && Mod(2, (ulong)(N + u), (ulong)p) != 1)**

**{**

**int n1 = p;**

**int countn1 = 0;**

**while (n1 > 0)**

**{**

**countn1++;**

**n1 /= 10;**

**}**

**if (similar.Contains(p))**

**{**

**countn1 = -1;**

**}**

**if (countn1 == size)**

**{**

**KG.Add(K);**

**pchG.Add(p);**

**countcikl++;**

**similar.Add(p);**

**}**

**break;**

**}**

**K++;**

**}**

**}**

**}**

**static void Main()**

**{**

**int size = 3;**

**List<int> prostch = new List<int>**

**{**

**2, 3, 5, 11, 13, 72, 167, 389, 173, 397, 179, 181, 401, 409, 191, 419, 193, 421, 17, 197, 431, 19, 199, 433, 23, 211, 439, 29, 223, 443, 31, 227, 449, 37, 229, 457, 41, 233, 461, 43, 239, 463, 47, 241, 467, 53, 251, 479, 59, 257, 487, 61, 263, 491, 67 ,269, 499, 71, 271,73, 277, 79, 281, 83, 283, 89, 293,97, 307, 101, 311, 103, 313,107, 317,109, 331, 113, 337, 127, 347, 131, 349, 137, 353,139, 359, 149, 367, 151, 373, 157, 379, 163, 383**

**};**

**List<int> pchM = new List<int>();**

**List<int> pchP = new List<int>();**

**List<int> pchG = new List<int>();**

**List<int> KM = new List<int>();**

**List<int> KP = new List<int>();**

**List<int> KG = new List<int>();**

**Console.WriteLine("Test Milera");**

**GenMiller(size, prostch, ref pchM, ref KM);**

**Console.Write("Num ");**

**for (int i = 1; i <= 10; i++)**

**{**

**Console.Write(i + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" p ");**

**foreach (int prch in pchM)**

**{**

**Console.Write(prch + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" K ");**

**foreach (int K in KM)**

**{**

**Console.Write(K + " ");**

**}**

**Console.WriteLine();**

**Console.WriteLine("Test Poklingtona");**

**GenPocklington(size, prostch, ref pchP, ref KP);**

**Console.Write("Num ");**

**for (int i = 1; i <= 10; i++)**

**{**

**Console.Write(i + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" p ");**

**foreach (int prch in pchP)**

**{**

**Console.Write(prch + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" K ");**

**foreach (int K in KP)**

**{**

**Console.Write(K + " ");**

**}**

**Console.WriteLine();**

**Console.WriteLine("GOST");**

**GenGOST(size, prostch, ref pchG, ref KG);**

**Console.Write("Num ");**

**for (int i = 1; i <= 10; i++)**

**{**

**Console.Write(i + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" p ");**

**foreach (int prch in pchG)**

**{**

**Console.Write(prch + " ");**

**}**

**Console.WriteLine();**

**Console.Write(" K ");**

**foreach (int K in KG)**

**{**

**Console.Write(K + " ");**

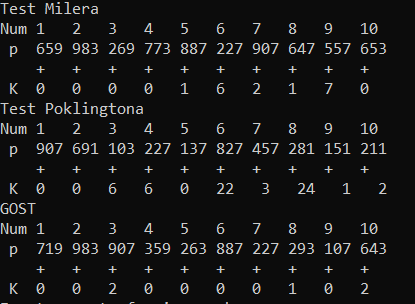
**}**

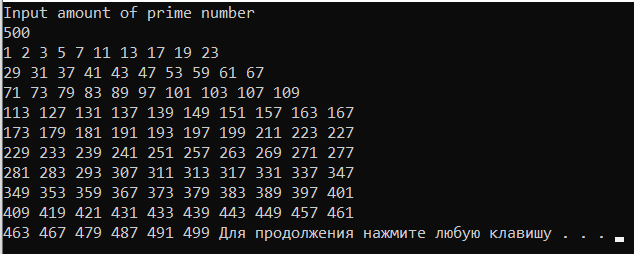
**Resheto();**

**}**

**}**

**Результат работы программы**

****

****

**Задание 3**

**C++**

**Листинг программы**

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include <iostream>

#include <vector>

using namespace std;

void cofe(int Tk, int Tsr, float r, vector<pair<float, float>>& xy) {//Функция считает x и y

for (float x = 0; x < 60; x++) {

float y = Tsr + (Tk - Tsr) / pow(M\_E, r \* x);

if (y < 0) break;

xy.push\_back({ x, y });

}

}

void aprox(vector<pair<float, float>> xy, float& a, float& b) {//Функция высчитывает значения апроксимирующей прямой

float sumx = 0;

float sumy = 0;

float sumxy = 0;

float sumx2 = 0;

for (int i = 0; i < 60; i++) {

sumx = sumx + xy[i].first;

sumy = sumy + xy[i].second;

sumxy = sumxy + xy[i].first \* xy[i].second;

sumx2 = sumx2 + xy[i].first \* xy[i].first;

}

a = (60 \* sumxy - sumx \* sumy) / (60 \* sumx2 - pow(sumx, 2));

b = (sumy - a \* sumx) / 60;

}

void korrel(vector<pair<float, float>> xy, float& kor) {//Функция высчитывает значение корреляции

float sredx = 0;

float sredy = 0;

for (pair<float, float> el : xy) {

sredx += el.first;

sredy += el.second;

}

sredx = sredx / 60;

sredy = sredy / 60;

float sumx = 0;

float sumy = 0;

float sumxy = 0;

for (pair<float, float> el : xy) {

sumx = sumx + pow(el.first - sredx, 2);

sumy = sumy + pow(el.second - sredy, 2);

sumxy = sumxy + (el.first - sredx) \* (el.second - sredy);

}

kor = sumxy / sqrt(sumx \* sumy);

}

int main() {

vector<pair<float, float>> xy;

cout << "Enter the coffee temperature" << endl;

int Tk;

cin >> Tk;

cout << "Enter the ambient temperature" << endl;

int Tsr;

cin >> Tsr;

cout << "Enter the cooling factor" << endl;

float r;

cin >> r;

cofe(Tk, Tsr, r, xy);

for (pair<float, float> el : xy) {

cout << el.first << " " << el.second << endl;

}

float a = 0;

float b = 0;

aprox(xy, a, b);

cout << "a = " << a << " b = " << b << endl;

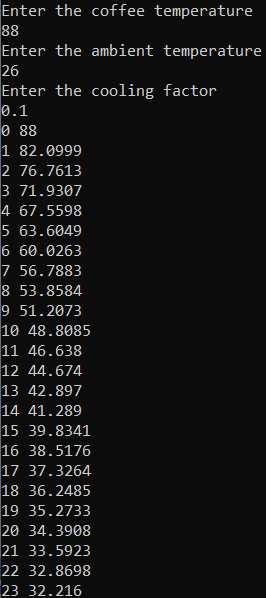
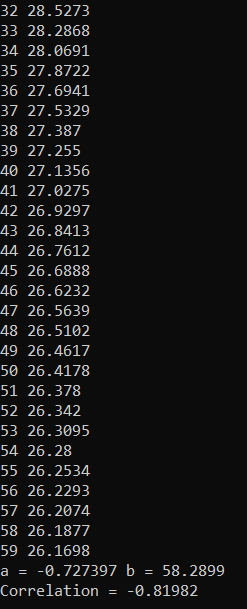
float kor = 0;

korrel(xy, kor);

cout << "Сorrelation = " << kor << endl;

}

**Результат работы программы**

**** ****

**С#**

**Листинг программы**

#define \_USE\_MATH\_DEFINES

using System;

using System.Collections.Generic;

namespace CoffeeTemperature

{

class Program

{

static void Cofe(int Tk, int Tsr, float r, List<(float, float)> xy)

{

for (float x = 0; x < 60; x++)

{

float y = Tsr + (Tk - Tsr) / (float)Math.Pow(Math.E, r \* x);

if (y < 0) break;

xy.Add((x, y));

}

}

static void Aprox(List<(float, float)> xy, out float a, out float b)

{

float sumx = 0;

float sumy = 0;

float sumxy = 0;

float sumx2 = 0;

for (int i = 0; i < 60; i++)

{

sumx += xy[i].Item1;

sumy += xy[i].Item2;

sumxy += xy[i].Item1 \* xy[i].Item2;

sumx2 += xy[i].Item1 \* xy[i].Item1;

}

a = (60 \* sumxy - sumx \* sumy) / (60 \* sumx2 - (float)Math.Pow(sumx, 2));

b = (sumy - a \* sumx) / 60;

}

static void Korrel(List<(float, float)> xy, out float kor)

{

float sredx = 0;

float sredy = 0;

foreach (var el in xy)

{

sredx += el.Item1;

sredy += el.Item2;

}

sredx /= 60;

sredy /= 60;

float sumx = 0;

float sumy = 0;

float sumxy = 0;

foreach (var el in xy)

{

sumx += (float)Math.Pow(el.Item1 - sredx, 2);

sumy += (float)Math.Pow(el.Item2 - sredy, 2);

sumxy += (el.Item1 - sredx) \* (el.Item2 - sredy);

}

kor = sumxy / (float)Math.Sqrt(sumx \* sumy);

}

static void Main(string[] args)

{

List<(float, float)> xy = new List<(float, float)>();

Console.WriteLine("Enter the coffee temperature");

int Tk = int.Parse(Console.ReadLine());

Console.WriteLine("Enter the ambient temperature");

int Tsr = int.Parse(Console.ReadLine());

Console.WriteLine("Enter the cooling factor");

float r = float.Parse(Console.ReadLine());

Cofe(Tk, Tsr, r, xy);

foreach (var el in xy)

{

Console.WriteLine($"{el.Item1} {el.Item2}");

}

float a = 0, b = 0;

Aprox(xy, out a, out b);

Console.WriteLine($"a = {a} b = {b}");

float kor = 0;

Korrel(xy, out kor);

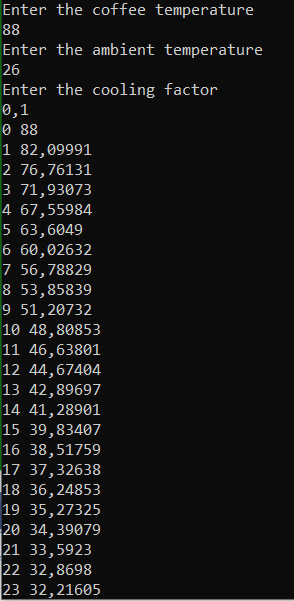
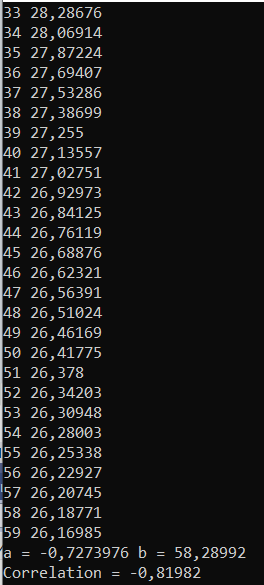
Console.WriteLine($"Correlation = {kor}");

}

}

}

**Результат работы программы**

**** ****

**Вывод**

В ходе выполнения лабораторной работы были реализованы циклические вычислительные процессы, позволяющие решать различные задачи.

В рамках первого задания была создана программа, вычисляющая значения функции, заданной графически, на заданном интервале с определённым шагом. Программа выводит результаты в виде таблицы с заголовком и шапкой, что позволяет наглядно представить полученные данные.

Второе задание включало реализацию тестов на простоту чисел, используя методы перебора делителей, тесты Миллера и Поклингтона, а также ГОСТ Р 34.10-94. Это позволило проверить числа на простоту и определить, являются ли они простыми.

Третье задание предполагало создание программы, моделирующей процесс остывания чашки кофе. Программа учитывает начальную температуру кофе, температуру окружающей среды и другие параметры, влияющие на скорость остывания. Результаты моделирования могут быть представлены в виде графиков или таблиц, что позволяет наглядно оценить динамику процесса.

Таким образом, выполнение лабораторной работы позволило закрепить навыки работы с циклическими процессами, научиться создавать программы для решения различных задач и анализировать полученные результаты.