МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации



**Лабораторная работа №3**

**по дисциплине: «Программирование»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-420», «АВТФ»,  *Иванеко Павел Константинович*  «30» апреля 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андрееввич*  «30» апреля 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
|  |  |

Новосибирск 2025

**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 на двух языках программирования (С++ и другой (по согласованию с преподавателем)).

**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.

**Задание 2.** Реализовать тесты на простоту согласно заданию.

**Задание 3.** Математическая сумма.

**Задание 4.** Увлекательная игра.

**Задание 5\*.** 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.

**Методика выполнения работы:**

1. определить типы используемых в программе данных;

2. описать переменные;

3. написать функции ввода-вывода;

4. разработать алгоритм решения задачи по индивидуальному заданию;

5. написать и отладить программу с вводом-выводом информации;

6. протестировать работу программы на различных исходных данных;

7. изменить формат вывода, проверить работу программы при другом формате вывода.

**Выполнение задания 1.**

**Листинг программы на С++**

#include <cmath>

#include<iostream>

#define X\_START -3

#define X\_FINITE 7

using namespace std;

void Input(float& X0,float& X1,float& dx)

{

cout << "Input X0: " << endl;

cin >> X0;

cout << "Input X1: " << endl;

cin >> X1;

cout << "Input dx: " << endl;

cin >> dx;

}

double func(float x)

{

if(x>=-3 && x<=-1)

return -x-1;

if(x>-1 && x <=1)

return 0;

if(x>1 && x<=5)

return sqrt(4-((x-3)\*(x-3)));

if(x>5 && x<=7)

return -x/2+2.5;

return 0;

}

void Output(float X0,float X1,float dx)

{

cout << "x" << "\t\t" << "y" << endl;

cout << endl;

for(float x = X0;x<=X1;x+=dx)

{

cout << x << "\t\t" << func(x) << endl;

}

}

int main()

{

float X0,X1,dx;

Input(X0,X1,dx);

if(dx<=0 || X0<X\_START || X1>X\_FINITE)

{

cout << "Error" << endl;

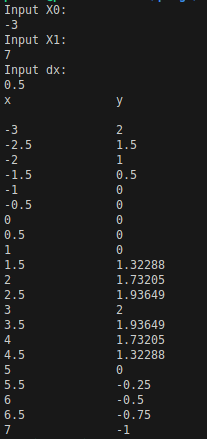
exit(1);

}

Output(X0,X1,dx);

}

**Результат работы программы представлен на рисунке 1.**

****

**Рисунок 1**

**Листинг программы на python**

import math

X\_START = -3

X\_FINITE = 7

def Input():

X0 = float(input("Input X0: "))

X1 = float(input("Input X1: "))

dx = float(input("Input dx: "))

return X0, X1, dx

def func(x):

if x >= -3 and x <= -1:

return -x - 1

if x > -1 and x <= 1:

return 0

if x > 1 and x <= 5:

return math.sqrt(4 - ((x - 3) \* (x - 3)))

if x > 5 and x <= 7:

return -x / 2 + 2.5

return 0

def Output(X0, X1, dx):

print("x\t\ty\n")

x = X0

while x <= X1:

print(f"{x}\t\t{func(x)}")

x += dx

def main():

X0, X1, dx = Input()

if dx <= 0 or X0 < X\_START or X1 > X\_FINITE:

print("Error")

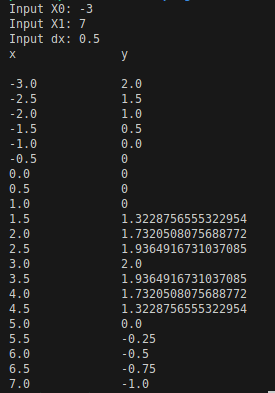
exit(1)

Output(X0, X1, dx)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 2.**

****

**Рисунок 2**

**Выполнение задания 2.**

**Тест Миллера.**

**Листинг программы на С++**

#include <iostream>

#include <vector>

#include <tuple>

#include <cmath>

#include <random>

#include <algorithm>

using namespace std;

// Глобальный генератор случайных чисел

mt19937 mt\_rand(random\_device{}());

vector<int> primes(int n);

pair<int, vector<int>> builder\_test(const vector<int>& prime, int bit);

int test\_millera(int n, int t, const vector<int>& q);

int power\_mod(int a, int b, int n);

int rn(int a, int b);

void print\_results(const vector<int>& res, const vector<string>& res\_ver\_test, const vector<int>& otvegnutie);

int main() {

int size\_primes = 500;

vector<int> prime = primes(size\_primes);

int bit;

cin >> bit;

if(bit<=0 || bit>=32)

{

cout<<"Error"<<endl;

exit(1);

}

vector<int> q;

int n;

int k = 0;

int probability;

vector<int> res;

vector<string> res\_ver\_test;

vector<int> otvegnutie;

while (res.size() != 10) {

tie(n, q) = builder\_test(prime, bit);

probability = test\_millera(n, 10, q);

if (probability == 1) {

if (find(res.begin(), res.end(), n) == res.end()) {

res.push\_back(n);

probability = test\_millera(n, 1, q);

res\_ver\_test.push\_back(probability == 1 ? "+" : "-");

otvegnutie.push\_back(k);

k = 0;

}

}

else {

k++;

}

}

print\_results(res, res\_ver\_test, otvegnutie);

}

// Решето Эратосфена

vector<int> primes(int n) {

vector<bool> is\_prime(n + 1, true);

vector<int> primes;

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

if (is\_prime[p]) {

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

is\_prime[i] = false;

}

}

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

if (is\_prime[p])

primes.push\_back(p);

}

return primes;

}

// Построение числа n

pair<int, vector<int>> builder\_test(const vector<int>& prime, int bit) {

int max\_index = 0;//индекс используемый для выбора простых чисел из prime

int max\_pow = 1;//степень до которой мы будем возводить простые числа из prime

while (max\_index < prime.size() && prime[max\_index] < pow(2, bit - 1)) max\_index++;

while (pow(2, max\_pow) < pow(2, bit - 1)) max\_pow++;

int m = 1;

vector<int> q;

while (true) {

int num = rn(0, max\_index - 1);

int power = rn(1, max\_pow);

if (pow(prime[num], power) != 0) { //проверка и добавление числа

if ((double)m \* pow(prime[num], power) < INT32\_MAX) {

m \*= pow(prime[num], power);

q.push\_back(prime[num]);

}

}

if (m > pow(2, bit - 2)) { //проверка размера числа

if (m >= pow(2, bit - 1)) {

m = 1;

q.clear();

} else {

break;

}

}

}

int n = 2 \* m + 1;

return make\_pair(n, q);

}

// Тест Миллера

int test\_millera(int n, int t, const vector<int>& q) {

vector<int> a;//числа для проверки 1<aj<n

while (a.size() != t) {

int aj = rn(2, n - 1);

if (find(a.begin(), a.end(), aj) == a.end()) {

a.push\_back(aj);

}

}

for (int aj : a) { //проверка степени числа

if (power\_mod(aj, n - 1, n) != 1) {

return 0;

}

}

for (int qi : q) {

bool flag = true;

for (int aj : a) {

if (power\_mod(aj, (n - 1) / qi, n) != 1) {

flag = false;

break;

}

}

if (flag) {

return 0;

}

}

return 1;

}

// Быстрое возведение в степень по модулю

int power\_mod(int a, int b, int n) {

long long result = 1;

long long base = a % n;

while (b > 0) {

if (b & 1) result = (result \* base) % n;

base = (base \* base) % n;

b >>= 1;

}

return static\_cast<int>(result);

}

// Глобальный рандомайзер

int rn(int a, int b) {

return uniform\_int\_distribution<int>(a, b)(mt\_rand);

}

// Печать результатов

void print\_results(const vector<int>& res, const vector<string>& res\_ver\_test, const vector<int>& otvegnutie) {

cout << "Prime Numbers\tTest Results\tOccurrences" << endl;

cout << "----------------------------------------------" << endl;

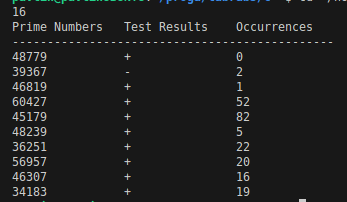
for (size\_t i = 0; i < res.size(); ++i) {

cout << res[i] << "\t\t" << res\_ver\_test[i] << "\t\t" << otvegnutie[i] << endl;

}

}

**Результат работы программы представлен на рисунке 3.**

****

**Рисунок 3**

**Листинг программы на python**

import random

import math

# Глобальный генератор случайных чисел

mt\_rand = random.Random()

def primes(n):

# Решето Эратосфена

is\_prime = [True] \* (n + 1)

primes = []

for p in range(2, int(n \*\* 0.5) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

for p in range(2, n + 1):

if is\_prime[p]:

primes.append(p)

return primes

def builder\_test(prime, bit):

# Построение числа n

max\_index = 0 # индекс используемый для выбора простых чисел из prime

max\_pow = 1 # степень до которой мы будем возводить простые числа из prime

while max\_index < len(prime) and prime[max\_index] < 2 \*\* (bit - 1):

max\_index += 1

while 2 \*\* max\_pow < 2 \*\* (bit - 1):

max\_pow += 1

m = 1

q = []

while True:

num = rn(0, max\_index - 1)

power = rn(1, max\_pow)

if pow(prime[num], power) != 0: # проверка и добавление числа

if m \* pow(prime[num], power) < (2 \*\* 31 - 1):

m \*= pow(prime[num], power)

q.append(prime[num])

if m > 2 \*\* (bit - 2): # проверка размера числа

if m >= 2 \*\* (bit - 1):

m = 1

q.clear()

else:

break

n = 2 \* m + 1

return n, q

def test\_millera(n, t, q):

# Тест Миллера

a = [] # числа для проверки 1<aj<n

while len(a) != t:

aj = rn(2, n - 1)

if aj not in a:

a.append(aj)

for aj in a: # проверка степени числа

if power\_mod(aj, n - 1, n) != 1:

return 0

flag = True

i = 0

for aj in a:

if i < len(q) and power\_mod(aj, (n - 1) // q[i], n) != 1:

flag = False

i += 1

if flag:

return 0

return 1

def power\_mod(a, b, n):

# Быстрое возведение в степень по модулю

result = 1

base = a % n

while b > 0:

if b & 1:

result = (result \* base) % n

base = (base \* base) % n

b >>= 1

return result

def rn(a, b):

# Глобальный рандомайзер

return mt\_rand.randint(a, b)

def print\_results(res, res\_ver\_test, otvegnutie):

# Печать результатов

print("Prime Numbers\tTest Results\tOccurrences")

print("----------------------------------------------")

for i in range(len(res)):

print(f"{res[i]}\t\t{res\_ver\_test[i]}\t\t{otvegnutie[i]}")

def main():

size\_primes = 500

prime = primes(size\_primes)

bit = int(input())

if bit <= 0 or bit >= 32:

print("Error")

exit(1)

q = []

res = []

res\_ver\_test = []

otvegnutie = []

k = 0

while len(res) != 10:

n, q = builder\_test(prime, bit)

probability = test\_millera(n, 10, q)

if probability == 1:

if n not in res:

res.append(n)

probability = test\_millera(n, 1, q)

res\_ver\_test.append("+" if probability == 1 else "-")

otvegnutie.append(k)

k = 0

else:

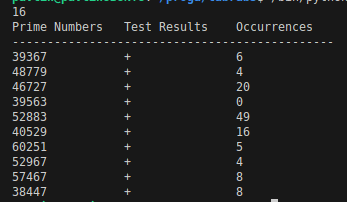
k += 1

print\_results(res, res\_ver\_test, otvegnutie)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 4.**



**Рисунок 4**

**Тест Поклингтона.**

**Листинг программы на С++**

#include <iostream>

#include <vector>

#include <tuple>

#include <cmath>

#include <random>

#include <algorithm>

using namespace std;

vector<int> primes(int n);

pair<int, vector<int>> builder\_test(vector<int> prime, int bit);

int test\_poklin(int n, int t, vector<int> q);

int power\_mod(int a, int b, int n);

int rn(int a, int b);

void print\_results(const vector<int>& res, const vector<string>& res\_ver\_test, const vector<int>& otvegnutie);

int main() {

int size\_primes = 500;

vector<int> prime = primes(size\_primes);

int bit = 0;

cin >> bit;

if(bit<=0 || bit>=32)

{

cout<<"error"<<endl;

exit(1);

}

vector<int> q;

int n;

int k = 0;

int probability;

vector<int> res;

vector<string> res\_ver\_test;

vector<int> otvegnutie;

while(res.size() != 10) {

tie(n, q) = builder\_test(prime, bit);

probability = test\_poklin(n, 10, q);

if(probability == 1) {

if (find(res.begin(), res.end(), n) == res.end()) {

res.push\_back(n);

probability = test\_poklin(n, 1, q);

if(probability == 1) {

res\_ver\_test.push\_back("+");

}

else{

res\_ver\_test.push\_back("-");

}

otvegnutie.push\_back(k);

k = 0;

}

}

else{

k++;

}

}

print\_results(res, res\_ver\_test, otvegnutie);

}

//решето Эратосфена

vector<int> primes(int n) {

vector<bool> is\_prime(n + 1, true);

vector<int> primes;

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

if (is\_prime[p]) {

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

is\_prime[i] = false;

}

}

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

if (is\_prime[p])

primes.push\_back(p);

}

return primes;

}

//строим число n=F\*R+1

pair<int, vector<int>> builder\_test(vector<int> prime, int bit) {

int max\_index = 0;

int max\_pow = 1;

for (; (prime[max\_index] < pow(2, (bit / 2) + 1)) && (max\_index < prime.size()); max\_index++);

for (; pow(2, max\_pow) < pow(2, (bit / 2) + 1); max\_pow++);

int f = 1;

vector<int> q;

while(true){

int num = rn(0, max\_index); //выбираем рандомный индекс

int power = rn(1, max\_pow); //выбираем рандомную степень

if (pow(prime[num], power)) {

if(f \* pow(prime[num], power) < INT32\_MAX) {

f \*= pow(prime[num], power);

q.push\_back(prime[num]);

}

}

if(f > pow(2, (bit / 2))){ //проверяем размер 2^(bit/2)<f<2^(bit/2)+1

if(f >= pow(2, (bit / 2) + 1)){

f = 1;

q.clear();

}

else{

break;

}

}

}

int R;

do

{

R = rn(pow(2, (bit / 2) - 1) + 1, pow(2, bit / 2)); //рандомное 2^(bit/2)-1<R<2^bit/2

} while (R % 2 != 0);

int n = R \* f + 1;

return make\_pair(n, q);

}

int test\_poklin(int n, int t, vector<int> q) {

vector<int> a;

int aj;

while (a.size() != t) {

aj = rn(1, n);

if (find(a.begin(), a.end(), aj) == a.end()) {

a.push\_back(aj);

}

}

//1ая проверка

for (int aj : a) {

if (power\_mod(aj, n - 1, n) != 1) {

return 0;

break;

}

}

//2ая проверка

for(int aj:a){

bool flag = true;

for (int qi : q) {

if (qi != 0 && power\_mod(aj, (n - 1) / qi, n) == 1) {

flag = false;

break;

}

}

if (flag) {

return 1;

}

}

return 0;

}

int power\_mod(int a, int b, int n) {

long long result = 1;

while (b > 0) {

if (b % 2 == 1)

result = (result \* a) % n;

a = (a \* a) % n;

b /= 2;

}

return result;

}

int rn(int a, int b) {

mt19937 mt\_rand(random\_device{}());

return uniform\_int\_distribution<int>(a, b)(mt\_rand);

}

void print\_results(const vector<int>& res, const vector<string>& res\_ver\_test, const vector<int>& otvegnutie) {

cout << "Prime Numbers\tTest Results\tOccurrences" << endl;

cout << "----------------------------------------------" << endl;

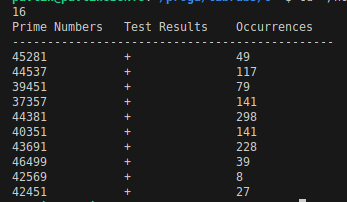
for (size\_t i = 0; i < res.size(); ++i) {

cout << res[i] << "\t\t" << res\_ver\_test[i] << "\t\t" << otvegnutie[i] << endl;

}

}

**Результат работы программы представлен на рисунке 5.**



**Рисунок 5**

**Листинг программы на python**

import math

import random

def primes(n):

is\_prime = [True] \* (n + 1)

result = []

for p in range(2, int(math.sqrt(n)) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

for p in range(2, n + 1):

if is\_prime[p]:

result.append(p)

return result

def power\_mod(a, b, n):

result = 1

a = a % n

while b > 0:

if b % 2 == 1:

result = (result \* a) % n

a = (a \* a) % n

b //= 2

return result

def rn(a, b):

return random.randint(a, b)

def builder\_test(prime, bit):

max\_index = 0

while max\_index < len(prime) and prime[max\_index] < 2 \*\* ((bit // 2) + 1):

max\_index += 1

max\_pow = 1

while 2 \*\* max\_pow < 2 \*\* ((bit // 2) + 1):

max\_pow += 1

f = 1

q = []

while True:

num = rn(0, max\_index - 1)

power = rn(1, max\_pow)

p\_pow = prime[num] \*\* power

if f \* p\_pow < 2\*\*31:

f \*= p\_pow

q.append(prime[num])

if f > 2 \*\* (bit // 2):

if f >= 2 \*\* ((bit // 2) + 1):

f = 1

q.clear()

else:

break

while True:

R = rn(2 \*\* ((bit // 2) - 1) + 1, 2 \*\* (bit // 2))

if R % 2 == 0:

break

n = R \* f + 1

return n, q

def test\_poklin(n, t, q):

a = []

while len(a) < t:

aj = rn(1, n - 1)

if aj not in a:

a.append(aj)

for aj in a:

if power\_mod(aj, n - 1, n) != 1:

return 0

for i, aj in enumerate(a):

if i < len(q) and q[i] != 0:

if power\_mod(aj, (n - 1) // q[i], n) == 1:

return 0

return 1

def print\_results(res, res\_ver\_test, otvegnutie):

print("Prime Numbers\tTest Results\tOccurrences")

print("----------------------------------------------")

for n, ver, k in zip(res, res\_ver\_test, otvegnutie):

print(f"{n}\t\t{ver}\t\t{k}")

def main():

size\_primes = 500

prime = primes(size\_primes)

bit = int(input())

if bit <= 0 or bit >= 32:

print("error")

return

res = []

res\_ver\_test = []

otvegnutie = []

k = 0

while len(res) != 10:

n, q = builder\_test(prime, bit)

probability = test\_poklin(n, 10, q)

if probability == 1 and n not in res:

res.append(n)

probability = test\_poklin(n, 1, q)

res\_ver\_test.append("+" if probability == 1 else "-")

otvegnutie.append(k)

k = 0

else:

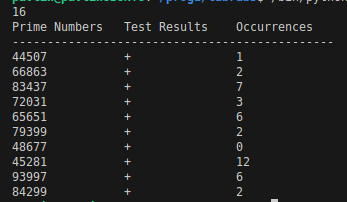
k += 1

print\_results(res, res\_ver\_test, otvegnutie)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 6.**



**Рисунок 6**

**Процедура генерации простых чисел ГОСТ Р 34.10-94**

**Листинг программы на С++**

#include <iostream>

#include <vector>

#include <cmath>

#include <random>

#include <algorithm>

using namespace std;

// решето эратосфена

vector<int> primes(int n);

int build\_new\_from\_old(vector<int> prime, int bit);

int power\_mod(int a, int b, int n);

int rn\_int(int a, int b);

double rn\_double(int a, int b);

void print\_res(vector<int> res);

int main() {

vector<int> prime = primes(500);

int bit;

cin >> bit;

if(bit<=0 || bit>=32)

{

cout<<"Error"<<endl;

exit(1);

}

vector<int> res;

while (res.size() != 10) {

int p = build\_new\_from\_old(prime, bit);

if (find(res.begin(), res.end(), p) == res.end()) {

res.push\_back(p);

}

}

print\_res(res);

}

vector<int> primes(int n) {

vector<bool> is\_prime(n + 1, true);

vector<int> primes;

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

if (is\_prime[p]) {

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

is\_prime[i] = false;

}

}

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

if (is\_prime[p])

primes.push\_back(p);

}

return primes;

}

//Алгоритм перехода от меньшего простого числа к большему

int build\_new\_from\_old(vector<int> prime, int bit) {

int q;

int max\_index = 0;

int p;

for (; prime[max\_index] < pow(2, bit / 2); max\_index++);

while (true) {

q = prime[rn\_int(0, max\_index)];

if (q > 0 && q <= pow(2, bit / 2) - 1) break;

}

while (true) {

double epsilon = rn\_double(0, 1);

double n0 = (pow(2, bit - 1) / q) + (pow(2, bit - 1) \* epsilon / q);

int n = ceil(n0);

if ((int)n % 2 == 1) n++;

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

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

if (p > pow(2, bit)) break;

if (power\_mod(2, p - 1, p) == 1 && power\_mod(2, n + u, p) != 1) return p;

}

}

}

int power\_mod(int a, int b, int n) {

int result = 1;

while (b > 0) {

if (b % 2 == 1)

result = (result \* a) % n;

a = (a \* a) % n;

b /= 2;

}

return result;

}

int rn\_int(int a, int b) {

static mt19937 mt\_rand(random\_device{}());

return uniform\_int\_distribution<int>(a, b)(mt\_rand);

}

double rn\_double(int a, int b) {

static mt19937 mt\_rand(random\_device{}());

return uniform\_real\_distribution<double>(a, b)(mt\_rand);

}

void print\_res(vector<int> res) {

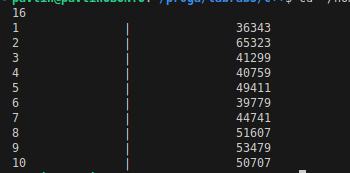
for (int i = 0; i < res.size(); i++) {

cout << i + 1 << "\t\t" << "|" << "\t\t" << res[i] << endl;

}

}

**Результат работы программы представлен на рисунке 7.**



**Рисунок 7**

**Листинг программы на python**

import math

import random

def primes(n):

is\_prime = [True] \* (n + 1)

result = []

for p in range(2, int(math.sqrt(n)) + 1):

if is\_prime[p]:

for i in range(p \* p, n + 1, p):

is\_prime[i] = False

for p in range(2, n + 1):

if is\_prime[p]:

result.append(p)

return result

def power\_mod(a, b, n):

result = 1

a = a % n

while b > 0:

if b % 2 == 1:

result = (result \* a) % n

a = (a \* a) % n

b //= 2

return result

def rn\_int(a, b):

return random.randint(a, b)

def rn\_double(a, b):

return random.uniform(a, b)

def build\_new\_from\_old(prime, bit):

max\_index = 0

while prime[max\_index] < 2 \*\* (bit // 2):

max\_index += 1

while True:

q = prime[rn\_int(0, max\_index - 1)]

if 2 \*\* ((bit // 2) - 1) < q <= 2 \*\* (bit // 2) - 1:

break

while True:

epsilon = rn\_double(0, 1)

n = int((2 \*\* (bit - 1) / q) + ((2 \*\* (bit - 1)) \* epsilon / q))

if n % 2 == 1:

n += 1

u = 0

while True:

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

if p > 2 \*\* bit:

break

if power\_mod(2, p - 1, p) == 1 and power\_mod(2, n + u, p) != 1:

return p

u += 2

def print\_res(res):

for i in range(len(res)):

print(str(i + 1) + "\t\t|\t\t" + str(res[i]))

def main():

prime = primes(500)

bit = int(input("Введите количество бит (1-18): "))

if bit <= 0 or bit >= 18:

print("Error")

return

res = []

while len(res) != 10:

p = build\_new\_from\_old(prime, bit)

if p not in res:

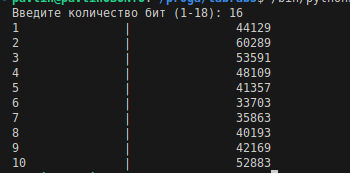
res.append(p)

print\_res(res)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 8.**



**Рисунок 8**

**Выполнение задания 3.**

**Листинг программы на С++**

#include <iostream>

#include <cmath>

#define epsilon 0.000001

#define limit 1000

#define max\_denominator 10000

int gcd(int a, int b) {

return (b == 0) ? a : gcd(b, a % b);

}

int is\_possibly\_rational(double x) {

for (int b = 1; b <= max\_denominator; b++) {

int a = (int)round(x \* b);

if (fabs(x - (double)a / b) < epsilon) {

int common\_divisor = gcd(a, b);

std::cout << a / common\_divisor << "/" << b / common\_divisor << std::endl;

return 1;

}

}

return -1;

}

void print(double num){

if (num == -1){

std::cout << "infinity" << std::endl;

}

else if (is\_possibly\_rational(num) == 1){

}

else{

std::cout << "irrational"<< std::endl;

}

}

void input(int& a,int& b){

std::cin >> a >> b;

}

double summing(int a,int b){

double cur;

double sum\_past = 0,sum = 0;

cur = 1./b;

sum\_past = cur;

sum = cur;

for (int n = 2;n<limit;n ++){

cur = pow(n,a)/pow(b,n);

sum += cur;

if (fabs(sum-sum\_past) < epsilon){

return sum;

}

sum\_past = sum;

}

return -1;

}

int main(void){

int a,b;

input(a,b);

double res = summing(a,b);

print(res);

}

**Результат работы программы представлен на рисунке 9.**



**Рисунок 9**

**Листинг программы на python**

import math

epsilon = 0.000001

limit = 1000

max\_denominator = 10000

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def is\_possibly\_rational(x):

for b in range(1, max\_denominator + 1):

a = round(x \* b)

if abs(x - a / b) < epsilon:

common\_divisor = gcd(a, b)

print(f"{a // common\_divisor}/{b // common\_divisor}")

return 1

return -1

def print\_result(num):

if num == -1:

print("infinity")

elif is\_possibly\_rational(num) == 1:

pass

else:

print("irrational")

def input\_values():

a, b = map(int, input().split())

return a, b

def summing(a, b):

cur = 1 / b

sum\_past = cur

sum\_value = cur

for n in range(2, limit):

cur = (n \*\* a) / (b \*\* n)

sum\_value += cur

if abs(sum\_value - sum\_past) < epsilon:

return sum\_value

sum\_past = sum\_value

return -1

def main():

a, b = input\_values()

res = summing(a, b)

print\_result(res)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 10.**



**Рисунок 10**

**Выполнение задания 4.**

**Листинг программы на С++**

#include <iostream>

#include <vector>

#include <algorithm>

#include <climits>

using namespace std;

int whoWins(int n, int m, const vector<int>& nums) {

// dp[i] — это максимальная разница очков (Павел - Вика),

vector<int> dp(n + 1, INT\_MIN);

dp[n] = 0; // если все числа уже удалены, разница очков = 0

//от "всё удалено" к "всё ещё осталось"

for (int i = n - 1; i >= 0; --i) {

int current\_sum = 0; // сумма чисел, которые мы возьмём на этом шаге

// Пробуем взять от 1 до m чисел если они остались

for (int k = 1; k <= m && i + k <= n; ++k) {

current\_sum += nums[i + k - 1];

// dp[i + k] — это разница очков, которую получит соперник после нас

// Т.е. забирает current\_sum, потом соперник играет с позиции i + k

dp[i] = max(dp[i], current\_sum - dp[i + k]);

}

}

return dp[0] > 0 ? 1 : 0;

}

void IO() {

int n, m;

cout << "Input n m :\n";

cin >> n >> m;

if(n<5 || n>50000 || m<4 || m>100)

{

cout<<"Error"<<endl;

exit(1);

}

vector<int> nums(n);

cout << "Input numbers:\n";

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

cin >> nums[i];

}

cout << whoWins(n, m, nums) << endl;

}

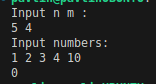
int main() {

IO();

return 0;

}

**Результат работы программы представлен на рисунке 11.**



**Рисунок 11**

**Листинг программы на python**

def who\_wins(n, m, nums):

# dp[i] — это максимальная разница очков (Павел - Вика),

dp = [float('-inf')] \* (n + 1)

dp[n] = 0 # если все числа уже удалены, разница очков = 0

# от "всё удалено" к "всё ещё осталось"

for i in range(n - 1, -1, -1):

current\_sum = 0 # сумма чисел, которые мы возьмём на этом шаге

# Пробуем взять от 1 до m чисел если они остались

for k in range(1, m + 1):

if i + k <= n:

current\_sum += nums[i + k - 1]

# dp[i + k] — это разница очков, которую получит соперник после нас

# Т.е. забирает current\_sum, потом соперник играет с позиции i + k

dp[i] = max(dp[i], current\_sum - dp[i + k])

return 1 if dp[0] > 0 else 0

def IO():

n, m = map(int, input("Input n m: ").split())

if not (5 <= n <= 50000) or not (4 <= m <= 100):

print("Error")

exit(1)

nums = list(map(int, input("Input numbers: ").split()))

print(who\_wins(n, m, nums))

def main():

IO()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 12.**



**Рисунок 12**

**Выполнение задания 5.**

**Листинг программы на С++**

#include<iostream>

#include<vector>

#include<cmath>

using namespace std;

double T;

double Ts;

double r;

int time\_limit;

pair<double,double> aproxCof;

double korrel(const vector<double>& temperatures,double mean\_y,int t)

{

double sum\_xy=0,sum\_x=0,sum\_y=0;

double mean\_x=(time\_limit-1)/2.0;

for(int x=0;x<=t;x++)

{

sum\_xy+=(x-mean\_x)\*(temperatures[x]-mean\_y);

sum\_x+=(x-mean\_x)\*(x-mean\_x);

sum\_y+=(temperatures[x]-mean\_y)\*(temperatures[x]-mean\_y);

}

return sum\_xy/ sqrt(sum\_x\*sum\_y);

}

pair<double,double> aprox(const vector<double>& x,const vector<double>& y)

{

int n = x.size();

double sum\_xy=0,sum\_x=0,sum\_y=0,sum\_x2=0;

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

{

sum\_xy+=x[i]\*y[i];

sum\_x+=x[i];

sum\_y+=y[i];

sum\_x2+=x[i]\*x[i];

}

double a = (n\*sum\_xy-sum\_x\*sum\_y)/(n\*sum\_x2-sum\_x\*sum\_x);

double b = (sum\_y-a\*sum\_x)/n;

pair<double,double> approx\_coef(0.0,0.0);

approx\_coef.first = a;

approx\_coef.second = b;

return approx\_coef;

}

vector<pair<double,double>> coffee(double T,double Ts,double r,double time\_limit)

{

vector<double> temperatures;

vector<double> time;

vector<pair<double,double>> temperatures\_corr;

for(int t=0;t<=time\_limit;t++)

{

temperatures.push\_back(Ts+(T-Ts)\*exp(-r\*t));

time.push\_back(t);

}

double mean\_y=0;

for(int i=0;i<=temperatures.size();i++)

{

mean\_y+=temperatures[i];

}

mean\_y=mean\_y/temperatures.size();

for(int t=0;t<=time\_limit;t++)

{

temperatures\_corr.push\_back({temperatures[t],korrel(temperatures,mean\_y,t)});

}

aproxCof=aprox(time,temperatures);

return temperatures\_corr;

}

int main()

{

cout<<"T: "<<endl;

cin>>T;

cout<<"Ts: "<<endl;

cin>>Ts;

cout<<"r: "<<endl;

cin>>r;

cout<<"time\_limit: "<<endl;

cin>>time\_limit;

vector<pair<double,double>> t\_c = coffee(T,Ts,r,time\_limit);

cout<<"коэффициенты аппроксимирующей прямой: a = "<< aproxCof.first<<"\t"<<"b = "<< aproxCof.second<<endl;;

cout<<"Время"<<"\t"<<"Температура"<<"\t"<<"Коэф Корреляции"<<endl;;

int t=0;

for(const auto& res:t\_c)

{

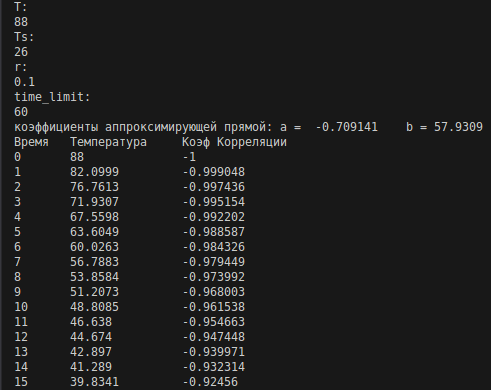
cout<<t<<"\t"<<res.first<<"\t\t"<<res.second<<endl;

t++;

}

}

**Результат работы программы представлен на рисунке 13.**



**Рисунок 13**

**Листинг программы на python**

import math

# Global variables

T = 0.0

Ts = 0.0

r = 0.0

time\_limit = 0

aproxCof = (0.0, 0.0)

def korrel(temperatures, mean\_y, t):

sum\_xy = 0

sum\_x = 0

sum\_y = 0

mean\_x = (time\_limit - 1) / 2.0

for x in range(t + 1):

sum\_xy += (x - mean\_x) \* (temperatures[x] - mean\_y)

sum\_x += (x - mean\_x) \*\* 2

sum\_y += (temperatures[x] - mean\_y) \*\* 2

return sum\_xy / math.sqrt(sum\_x \* sum\_y)

def aprox(x, y):

n = len(x)

sum\_xy = 0

sum\_x = 0

sum\_y = 0

sum\_x2 = 0

for i in range(n):

sum\_xy += x[i] \* y[i]

sum\_x += x[i]

sum\_y += y[i]

sum\_x2 += x[i] \*\* 2

a = (n \* sum\_xy - sum\_x \* sum\_y) / (n \* sum\_x2 - sum\_x \* sum\_x)

b = (sum\_y - a \* sum\_x) / n

return a, b

def coffee(T, Ts, r, time\_limit):

temperatures = []

time = []

temperatures\_corr = []

for t in range(time\_limit + 1):

temperatures.append(Ts + (T - Ts) \* math.exp(-r \* t))

time.append(t)

mean\_y = sum(temperatures) / len(temperatures)

for t in range(time\_limit + 1):

temperatures\_corr.append((temperatures[t], korrel(temperatures, mean\_y, t)))

global aproxCof

aproxCof = aprox(time, temperatures)

return temperatures\_corr

def main():

global T, Ts, r, time\_limit

T = float(input("T: "))

Ts = float(input("Ts: "))

r = float(input("r: "))

time\_limit = int(input("time\_limit: "))

t\_c = coffee(T, Ts, r, time\_limit)

print(f"aprox coaf: a = {aproxCof[0]}\tb = {aproxCof[1]}")

print("Время\tТемпература\t\tКоэф Корреляции")

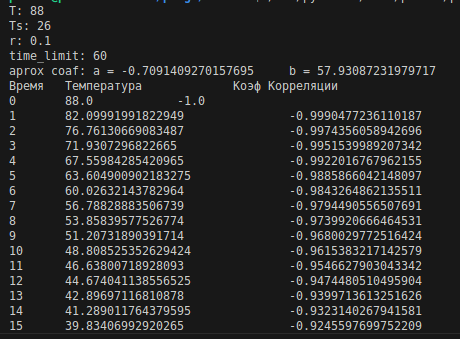
for t, (temperature, correlation) in enumerate(t\_c):

print(f"{t}\t{temperature}\t\t{correlation}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат работы программы представлен на рисунке 14.**



**Рисунок 14.**

**Вывод:** В ходе лабораторной работы были выполнены все поставленные цели и задачи, а именно: изучение циклических алгоритмов, операторов цикла, освоено программирование циклического вычислительного процесса. Был реализован циклический вычислительный процесс, а также решены задачи.