МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**ОТЧЁТ**

**по практической работе №2**

«Изучение математических основ криптографии»

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

Выполнил:Проверил:

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

*Савицкий Н. А. Архипова А. Б.*

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

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

(подпись) (подпись)

Новосибирск 2024

**Цели и задачи работы**: изучение циклических алгоритмов, операторов цикла, программирование циклического вычислительного процесса на примере математических методов основ криптографии.

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

**Задание 1.** Реализовать *ax* mod *p* сравнения по модулю простого числа через теорему Ферма и свойства сравнений. Программа должна проверять условия выполнения теоремы Ферма и простоту вводимого пользователем модуля. Реализовать алгоритм через разложение степени в двоичный вид (логарифм).

**Задание 2.** Реализовать обобщенный алгоритм Евклида для вычисления *с*\**d* mod *m*=1.

**Задание 3.** Реализовать расширенный алгоритм Евклида для вычисления взаимообратного числа *с-1* mod *m* = *d*.

**Задание 4**. Написать программу, использующую алгоритм шифрования данных для преобразования исходного текста.

Вариант 3 - Эль-Гамаля.

**Задание 5.** Найти последнюю цифру «трехэтажного числа».

Например, 3 7^8.

**Задание 6**. Написать сообщение на тему «Атака посередине». Можно продемонстрировать эмуляцию атаки на базе программы задания 4 данной практической работы.

**Задание 7**. Написать сообщение на тему «Стандарты современной криптографии в РФ».

**Методика выполнения работы:**

1. Разработать алгоритм решения задачи по индивидуальному заданию.

2. Написать и отладить программу решения задачи.

3. Протестировать работу программы на различных исходных данных.

4. Ответить на вопросы по выполненным заданиям, по запросу преподавателя модифицировать код.

5. Ответить на вопросы согласно списку понятий к защите практики (с численными примерами).

6. По запросу преподавателя решить практическое задание на тему «Изучение математических основ криптографии».

**Программы решения задачи на языке C++ с их результатами работы.**

**Задание 1.**

***ax* mod *p* сравнения по модулю простого числа через теорему Ферма и свойства сравнений.**

#include <iostream>

#include <cmath>

using namespace std;

bool isPrime(int n) //функция определения простого числа

{

for (int i = 2; i <= sqrt(n); i++)

{

if (n % i == 0)

return false;

}

return true;

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << "Введите основание: ";

int a;

cin >> a;

cout << "Введите модуль: ";

int p;

cin >> p;

cout << "Введите степень: ";

int degree;

cin >> degree;

int result = 1;

if (isPrime(p)) { //проверка модуля на простоту

if (a % p == 0) { //проверка условия a mod p != 0

cout << "Условие теоремы Ферма не выполнено!";

} else {

for (int i = 1; i <= degree; i++) {

result = (result \* a) % p;

}

cout << "Результат: " << result;

}

} else { //модуль не является простым числом

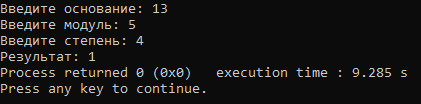
cout << "Модуль не является простым числом!";

}

return 0;

}

**Результат выполнения**



**Алгоритм через разложение степени в двоичный вид**

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

bool isPrime(int n) {

for (int i = 2; i <= sqrt(n); i++) {

if (n % i == 0)

return false;

}

return true;

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << "Введите основание: ";

int a;

cin >> a;

cout << "Введите модуль: ";

int p;

cin >> p;

cout << "Введите степень: ";

int degree;

cin >> degree;

int result = 1;

if (isPrime(p)) {

if (a % p == 0) {

cout << "Условие теоремы Ферма не выполнено!";

} else {

int logDegree = log2(degree);

int j = logDegree;

vector<int> binary(32, 0); // Инициализация вектора с 32 элементами

for (int i = logDegree; i >= 0; i--) {

int bit = ((degree >> i) & 1);

binary[j] = bit;

j--;

}

result = 1;

for (int i = logDegree; i >= 0; i--) {

result = (result \* result) % p;

if (binary[i] == 1) {

result = (result \* a) % p;

}

}

cout << "Результат: " << result << endl;

}

} else {

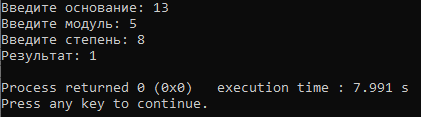
cout << "Модуль не является простым числом!";

}

return 0;

}

**Результат выполнения**



**Задание 2**

#include <iostream>

using namespace std;

int genEuclid(int a, int b) { //Обобщенный алгоритм Евклида

int r;

while (b != 0) {

r = a % b;

a = b;

b = r;

}

return a;

}

int numReverse(int c, int m) { // нахождение обратного

if(genEuclid(c, m) != 1) {

cout << "Такого не существует!";

return -1;

}

for (int d = 1; d < m; d++) {

if ((c\*d)%m == 1) {

return d;

}

}

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << "Введите c: ";

int c;

cin >> c;

cout << "Введите m: ";

int m;

cin >> m;

int d = numReverse(c, m);

if (d != -1) {

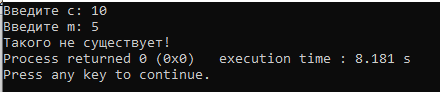
cout << "Число d равно: " << d;

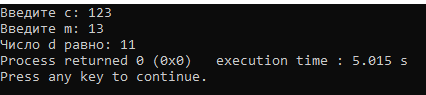
}

return 0;

}

**Результат выполнения**

****

****

**Задание 3**

#include <iostream>

using namespace std;

int advEuclid(int a, int b, int &u, int &v) { // Расширенный Евклид

if (b == 0) {

u = 1;

v = 0;

return a;

}

int u1, v1;

int nod = advEuclid(b, a % b, u1, v1);

u = v1;

v = u1 - a / b \* v1;

return nod;

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << "Введите c: ";

int c;

cin >> c;

cout << "Введите m: ";

int m;

cin >> m;

int u, v;

int nod = advEuclid(c, m, u, v);

if (nod != 1) {

cout << "Не существует взаимнообратного числа.";

return 1;

}

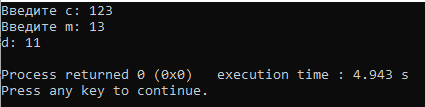
int d = (u % m + m) % m;

cout << "d: " << d << endl;

return 0;

}

**Результат выполнения**

****

**Задание 4**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <vector>

using namespace std;

bool IsPrime(long long n)

{

if (n <= 1)

return false;

for (long long i = 2; i \* i <= n; i++)

{

if (n % i == 0)

return false;

}

return true;

}

long long InDegreeByMod(long long a, long long b, long long m)

{

long long result = 1;

a = a % m;

while (b > 0)

{

if (b % 2 == 1)

result = (result \* a) % m;

b = b >> 1;

a = (a \* a) % m;

}

return result;

}

long long FindG(long long p)

{

if (!IsPrime(p))

{

cout << "Число p не является простым!" << endl;

return -1;

}

srand(time(NULL));

long long gRand = rand() % (p - 1) + 1; // Генерируем случайное число для g

while (true)

{

bool primSqrt = true;

for (long long i = 2; i < p - 1; i++)

{

if (InDegreeByMod(gRand, i, p) == 1)

{

primSqrt = false;

break;

}

}

if (primSqrt)

return gRand;

gRand = rand() % (p - 1) + 1; // Генерируем новое случайное число для g

}

return -1; // Если не удалось найти примитивный корень

}

long long GenSecKey(long long n)

{

srand(time(NULL));

return rand() % (n - 1) + 1; // Генерация случайного числа x из диапазона [1, n-1]

}

pair<long long, long long> GeneratePublicKey(long long p, long long g, long long x)

{

long long y = InDegreeByMod(g, x, p); // Вычисление открытого ключа y

return make\_pair(p, y);

}

pair<long long, long long> ForEncrypt(long long m, long long p, long long g, long long y)

{

srand(time(NULL));

long long k = rand() % (p - 1) + 1; // Генерация случайного числа k из диапазона [1, p-1]

long long c1 = InDegreeByMod(g, k, p);

long long c2 = (m \* InDegreeByMod(y, k, p)) % p;

return make\_pair(c1, c2);

}

long long Decrypt(pair<long long, long long> txtEncrypt, long long p, long long x)

{

long long c1 = txtEncrypt.first;

long long c2 = txtEncrypt.second;

long long m = (c2 \* InDegreeByMod(c1, p - 1 - x, p)) % p;

return m;

}

int main()

{

setlocale(LC\_ALL, "Russian");

cout << "Выберите действие:" << endl;

cout << "1. Зашифровать сообщение" << endl;

cout << "2. Расшифровать сообщение" << endl;

cout << "Ваш выбор: ";

int choice;

cin >> choice;

if (choice == 1)

{

cout << "Введите простое число p: ";

long long p;

cin >> p;

long long g = FindG(p);

if (g == -1)

{

cout << "Не найдено примитивного корня!" << endl;

return 0;

}

cout << "Найденный примитивный корень g: " << g << endl;

long long x = GenSecKey(p - 1);

cout << "Секретный ключ x: " << x << endl;

pair<long long, long long> publicKey = GeneratePublicKey(p, g, x);

cout << "Открытый ключ (p, g, y): (" << publicKey.first << ", " << g << ", " << publicKey.second << ")" << endl;

cout << "Введите сообщение m для шифрования: ";

long long m;

cin >> m;

pair<long long, long long> txtEncrypt = ForEncrypt(m, publicKey.first, g, publicKey.second);

cout << "Шифртекст (c1, c2): (" << txtEncrypt.first << ", " << txtEncrypt.second << ")" << endl;

}

else if (choice == 2)

{

cout << "Введите закрытый ключ x: ";

long long x;

cin >> x;

cout << "Введите простое число p: ";

long long p;

cin >> p;

cout << "Введите примитивный корень g: ";

long long g;

cin >> g;

cout << "Введите шифртекст (c1, c2): ";

long long c1, c2;

cin >> c1 >> c2;

long long decrMsg = Decrypt(make\_pair(c1, c2), p, x);

cout << "Расшифрованное сообщение: " << decrMsg << endl;

}

else

{

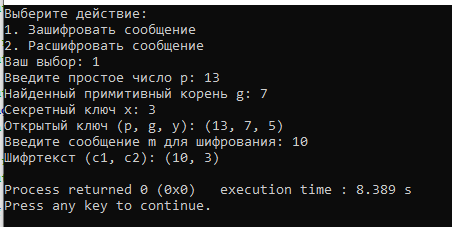
cout << "Неверный выбор. Пожалуйста, выберите 1 или 2." << endl;

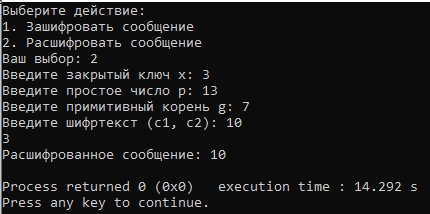
}

return 0;

}

**Результат выполнения**

****

****

**Задание 5**

#include <iostream>

using namespace std;

int lastNum(int num, int degree) {

if (degree == 0) return 1; // Любое число в степени 0 равно 1

num = num % 10; // Переводим основание в единицы

// Находим остаток от деления на 4, т.к. период длины 4 для последних цифр возведений чисел в степень

int ostatok = degree % 4;

if (ostatok == 0) ostatok = 4; // Корректируем для четырехстепенного цикла

int result = 1;

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

result = (result \* num) % 10; // Находим последнюю цифру возведения числа в степень

}

return result;

}

int main() {

setlocale(LC\_ALL, "Russian");

cout << "Введите число: ";

int num;

cin >> num;

cout << "Введите первую степень: ";

int degree1;

cin >> degree1;

cout << "Введите вторую степень: ";

int degree2;

cin >> degree2;

int finalDegree = lastNum(degree1, degree2);

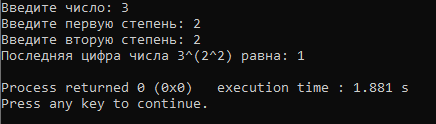
int result = lastNum(num, finalDegree);

cout << "Последняя цифра числа " << num << "^(" << degree1 << "^" << degree2 << ") равна: " << result << endl;

return 0;

}

**Результат выполнения**

****

**Программы решения задачи на языке Rust с их результатами работы.**

**Задание 1.**