МЕНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №3

Криптоаналіз афінної біграмної підстановки

Виконали:
ФБ-33 Охріменко Анастасія

ФБ-33 Телегіна Софія

Перевірила:
Селюх Поліна Валентинівна

Київ - 2025

Мета роботи:

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Порядок виконання роботи

1. **Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв'язуванням лінійних порівнянь. При розв'язуванні порівнянь потрібно коректно обробляти випадок із декількома розв'язками, повертаючи їх усі.**

```cpp
// ------------------ Розширений алгоритм Евкліда ------------------
tuple<ll, ll, ll> euclidean_algorithm(ll a, ll b) {
    if (a == 0) return { b, 0, 1 };
    auto [g, u1, v1] = euclidean_algorithm(b % a, a);
    ll u = v1 - (b / a) * u1;
    ll v = u1;
    return { g, u, v };
}

// ------------------ Розв'язок лінійного порівняння ------------------
vector<ll> modular_linear_equation_solver(ll a, ll b, ll n) {
    a = ((a % n) + n) % n;
    b = ((b % n) + n) % n;
    auto [d, u, v] = euclidean_algorithm(a, n);
    vector<ll> roots;
    if (b % d != 0) return roots;

    ll x0 = (u * (b / d)) % n;
    if (x0 < 0) x0 += n;

    for (ll i = 0; i < d; i++) {
        roots.push_back((x0 + i * (n / d)) % n);
    }
    return roots;
}
```

euclidean_algorithm - розширений алгоритм Евкліда (використовується при розв'язку рівняння). Знаходить такі числа g, u, v, що:

g=gcd(a,b)

a·u+b·v=g

modular_linear_equation_solver - розв'язок рівняння $ax \equiv b (mod\, n)$

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп'ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

```cpp
// ------------------ Номер біграми ------------------
int get_bigram_id(const wstring& bigram) {
    int M = ALPHABET.size();
    int i1 = ALPHABET.find(bigram[0]);
    int i2 = ALPHABET.find(bigram[1]);
    return i1 * M + i2;
}

// ------------------ Обчислення частот біграм ------------------
map<wstring, double> calculate_frequencies(const wstring& text) {
    map<wstring, double> freq;
    int bigram_count = 0;
    for (size_t i = 0; i + 1 < text.size(); i += 2) {
        wstring bg = text.substr(i, 2);
        freq[bg]++;
        bigram_count++;
    }
    for (auto& [k, v] : freq) v /= bigram_count;
    return freq;
}
```

get_bigram_id - перетворює біграму (2 символи) у унікальний числовий індекс

calculate_frequencies - обчислює частоти всіх біграм у тексті

3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п’яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ (a,b) шляхом розв’язання системи (1).

```cpp
// ------------------ Генерація ключів для пари біграм ------------------
vector<pair<ll, ll>> generate_key(const wstring& p1, const wstring& p2) {
    int M = ALPHABET.size();
    ll N = 1LL * M * M;

    ll x1 = get_bigram_id(p1.substr(0, 2));
    ll y1 = get_bigram_id(p1.substr(2, 2));
    ll x2 = get_bigram_id(p2.substr(0, 2));
    ll y2 = get_bigram_id(p2.substr(2, 2));

    vector<ll> roots = modular_linear_equation_solver(x1 - x2, y1 - y2, N);
    vector<pair<ll, ll>> keys;
    for (ll a : roots) {
        ll b = (y1 - a * x1) % N;
        if (b < 0) b += N;
        keys.emplace_back(a, b);
    }
    return keys;
}

// ------------------ Генерація всіх ключів ------------------
set<pair<ll, ll>> generate_keys(const vector<wstring>& open_bigrams,
    const vector<wstring>& cipher_bigrams) {
    set<pair<ll, ll>> keys;
    for (auto& p1 : open_bigrams) {
        for (auto& c1 : cipher_bigrams) {
            for (auto& p2 : open_bigrams) {
                for (auto& c2 : cipher_bigrams) {
                    if (p1 == p2 || c1 == c2) continue;
                    ll x1 = get_bigram_id(p1);
                    ll x2 = get_bigram_id(p2);
                    ll y1 = get_bigram_id(c1);
                    ll y2 = get_bigram_id(c2);
                    ll M = ALPHABET.size();
                    ll N = 1LL * M * M;
                    auto roots = modular_linear_equation_solver((x1 - x2 + N) % N, (y1 - y2 + N) % N, N);
                    for (ll a : roots) {
                        ll b = (y1 - a * x1) % N;
                        if (b < 0) b += N;
                        keys.insert({ a, b });
                    }
                }
            }
        }
    }
    return keys;
}
```

generate_key - знаходить список усіх можливих ключів (a, b). Зчитує значення x1, y1, x2, y2 - це числові представлення біграм:

x1, x2 - біграми відкритого тексту;

y1, y2 - біграми шифртексту.

Знаходить усі можливі a, які задовольняють: $(x1 - x2) \cdot a \equiv (y1 - y2)\ (mod N)$

Для кожного a знаходить b: $b = y1 - a \cdot x1\ (mod N)$

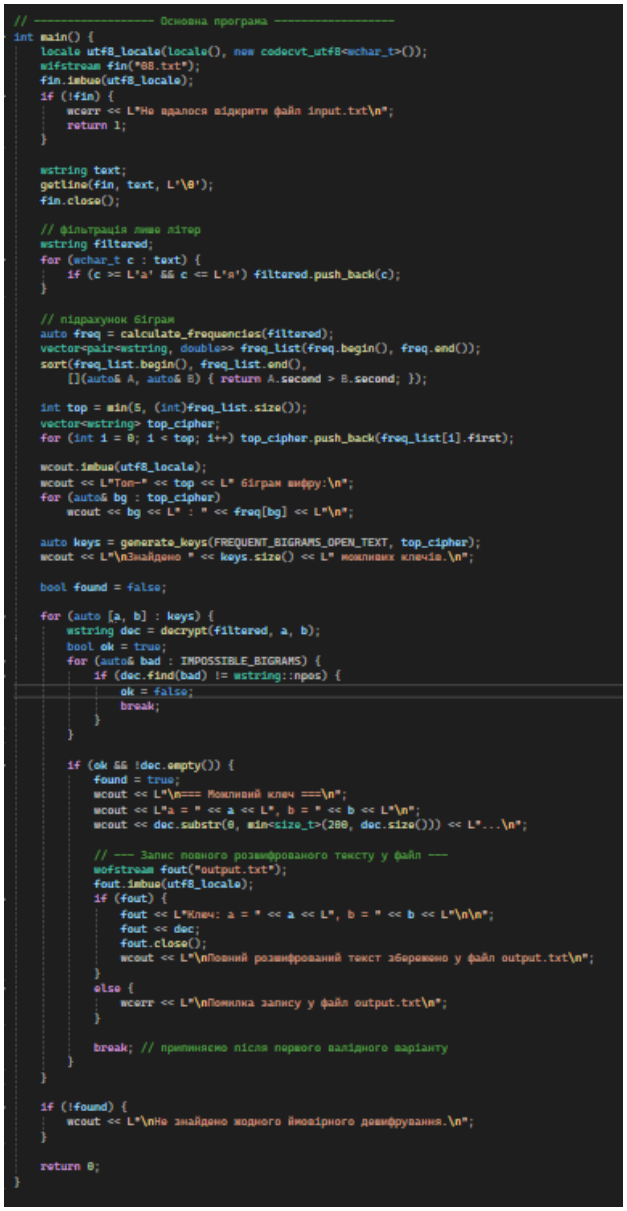
generate_keys - генерує всі можливі ключі (a, b), використовуючи всі комбінації двох біграм з відкритого тексту і шифртексту.

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата

```
// ------------------ Дешифрування ------------------
wstring decrypt(const wstring& text, ll a, ll b) {
    int M = ALPHABET.size();
    ll N = 1LL * M * M;
    auto [g, inv_a, tmp] = euclidean_algorithm(a, N);
    if (g != 1) return L"";

    inv_a = (inv_a % N + N) % N;

    wstring res;
    for (size_t i = 0; i + 1 < text.size(); i += 2) {
        wstring bg = text.substr(i, 2);
        if (ALPHABET.find(bg[0]) == wstring::npos || ALPHABET.find(bg[1]) == wstring::npos)
            return L"";
        ll y = get_bigram_id(bg);
        ll x = (inv_a * (y - b)) % N;
        if (x < 0) x += N;
        res.push_back(ALPHABET[x / M]);
        res.push_back(ALPHABET[x % M]);
    }
    return res;
}
```



```
// ------------------ Основна програма ------------------
int main() {
    locale utf8_locale(locale(), new codecvt_utf8<wchar_t>());
    wifstream fin("88.txt");
    fin.imbue(utf8_locale);
    if (!fin) {
        wcerr << L"Не вдалося відкрити файл input.txt\n";
        return 1;
    }

    wstring text;
    getline(fin, text, L'\0');
    fin.close();

    // фільтрація лише літер
    wstring filtered;
    for (wchar_t c : text) {
        if (c >= L'а' && c <= L'я') filtered.push_back(c);
    }

    // підрахунок біграм
    auto freq = calculate_frequencies(filtered);
    vector<pair<wstring, double>> freq_list(freq.begin(), freq.end());
    sort(freq_list.begin(), freq_list.end(),
        [](auto& A, auto& B) { return A.second > B.second; });

    int top = min(5, (int)freq_list.size());
    vector<wstring> top_cipher;
    for (int i = 0; i < top; i++) top_cipher.push_back(freq_list[i].first);

    wcout.imbue(utf8_locale);
    wcout << L"Топ-" << top << L" біграм шифру:\n";
    for (auto& bg : top_cipher)
        wcout << bg << L" : " << freq[bg] << L"\n";

    auto keys = generate_keys(FREQUENT_BIGRAMS_OPEN_TEXT, top_cipher);
    wcout << L"\nЗнайдено " << keys.size() << L" можливих ключів.\n";

    bool found = false;

    for (auto [a, b] : keys) {
        wstring dec = decrypt(filtered, a, b);
        bool ok = true;
        for (auto& bad : IMPOSSIBLE_BIGRAMS) {
            if (dec.find(bad) != wstring::npos) {
                ok = false;
                break;
            }
        }

        if (ok && !dec.empty()) {
            found = true;
            wcout << L"\n=== Можливий ключ ===\n";
            wcout << L"a = " << a << L", b = " << b << L"\n";
            wcout << dec.substr(0, min<size_t>(200, dec.size())) << L"...\n";

            // --- Запис повного розшифрованого тексту у файл ---
            wofstream fout("output.txt");
            fout.imbue(utf8_locale);
            if (fout) {
                fout << L"Ключ: a = " << a << L", b = " << b << L"\n\n";
                fout << dec;
                fout.close();
                wcout << L"\nПовний розшифрований текст збережено у файл output.txt\n";
            }
            else {
                wcerr << L"\nПомилка запису у файл output.txt\n";
            }

            break; // припиняємо після першого валідного варіанту
        }
    }

    if (!found) {
        wcout << L"\nНе знайдено жодного ймовірного дешифрування.\n";
    }

    return 0;
}
```

decrypt - функція для дешифрування біграмного афінного шифру: $x \equiv a^{-1}(y - b)(mod\ N)$

main - зчитує зашифрований текст, підраховує частоту біграм та сортує за спаданням частоти, дешифрує текст, перевіряє, чи в результаті немає неможливих біграм (типу ыы, аь, оы тощо), виводить найможливіший варіант. Запис повного розшифрованого тексту у файл

Результат виконання:

```
Топ-5 біграм шифру:
жц : 0.0145414
дэ : 0.0138702
цэ : 0.0134228
сц : 0.0127517
оц : 0.0123043

Знайдено 392 можливих ключів.

=== Можливий ключ ===
a = 17, b = 94
мальчикизаулыбалисьисжаромвзялисьзаделоонирвализолотистыецветыцветычтонаводняютвесьмирпереплескиваютсяслужаекнамощеныеул
ицытихонькостучатсявпрозрачныеокнапогребовнезнаютугомонуиудержуивсевокругзаливаю...

Повний розшифрований текст збережено у файл output.txt
```

Мальчикизаулыбалисьисжаромвзялисьзаделоонирвализолотистыецветыцветычтонавод
няютвесьмирпереплескиваютсяслужаекнамощеныеулицытихонькостучатсявпрозрачны
еокнапогребовнезнаютугомонуиудержуивсевокругзаливаютслепящимсверканиемраспл
авленногосолнцакаждоелетоониточносцеписрываютсясказалдедушкапустьихянепротив
вонихсколькостоятгордыекакльвыпосмотришьнанихподольшетакипрожгутутебявглаза
хдыркуведьпростойцветокможносказатьсорнаятраваниктоеенезамечаетамыуважаемсчи
таемодуванчикблагородноерастениеонинабралиполныемешкиодуванчиковиунесливниз
впогребвывалилиихизмешковивотьмепогребаразлилосьсияниевинныйпрессдожидалсяи
хоткрытыйхолодныйзолотистыйпотоксогрелегодедушкапередвинулпрессповернулручк
узавертелбыстрейбыстрейипрессмягкостиснулдобычунувотвоттаксперватонкойструйк
ойпотомвсещедреебильнеепобежалпожелобувглиняныекувшинысокпрекрасногожарк
огомесяцаемудалиперебродитьснялипенуиразлиливчистыебутылкиизподкетчупаионив
ыстроилисьрядаминаполкахпоблескиваявсумракепогребавиноизодуванчиковсамыеэтис
ловаточнолетонаязыкевиноизодуванчиковпойманноеизакупоренноевбутылкилетоитепе
рькогдадугласзналпонастоящемузналчтоонживойчтоонзатемиходитпоземлечтобывидет
ьиощущатьмироннопонялещеоднонадочастицувсегочтоонузналчастицуэтогоособенногод
няднясбораодуванчиковтожезакупоритьисохранитьапотомнастанеттакойзимнийянварс
кийденькогдавалитгустойснегисолнцаужедавнымдавноникто невиделиможетбытьэточу
допозабылосьихорошобыегосновавспомнитьвоттогдаонегооткупоритведьэтолетонепре
меннобудетлетомнежданныхчудесинадовсеихсберечьигдетоотложитьдлясебячтобыпос
левлюбойчаскогдавздумаешьпробратьсянацыпочкахвовлажныйсумракипротянутьруку
итамрядзарядомбудутстоятьбутылкисвиномизодуванчиковонобудетмягкомерцатьточн
ораскрывающиесяназарецветыасквозьтонкийслойпылибудетпоблескиватьсолнценыне
шнегоиюнявзглянисквозьэтовинонахолодныйзимнийденьиснеграстаетизподнегопокаж
етсятраванадере

Висновок:

Було розглянуто алгоритм афінного шифру та розшифровано текст за допомогою зворотнього алгоритму, найчастіше зустрічаємих біграм в шифротексті та у мові відкритого тексту