

Информационная безопасность.

Лабораторная работа №8.

Zayan Ahamed Ally

Содержание

1	Цель работы	1
2	Задание	1
3	Выполнение лабораторной работы.....	1
4	Выводы.....	4

1 Цель работы

Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом

2 Задание

Написать код для шифровки и дешифровки сообщений.

3 Выполнение лабораторной работы

Написал код для зашифровки и дешифровки кодов. Вывод программы. Для подробных объяснений обратитесь к скринкасту на youtube.

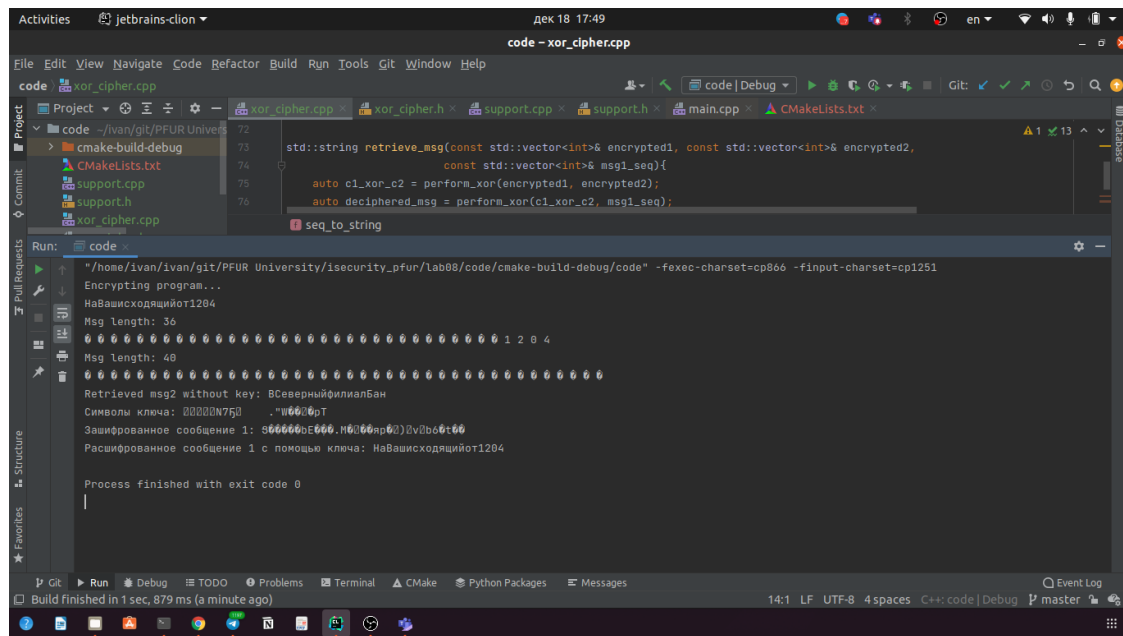


Figure 1: Вывод

Код:

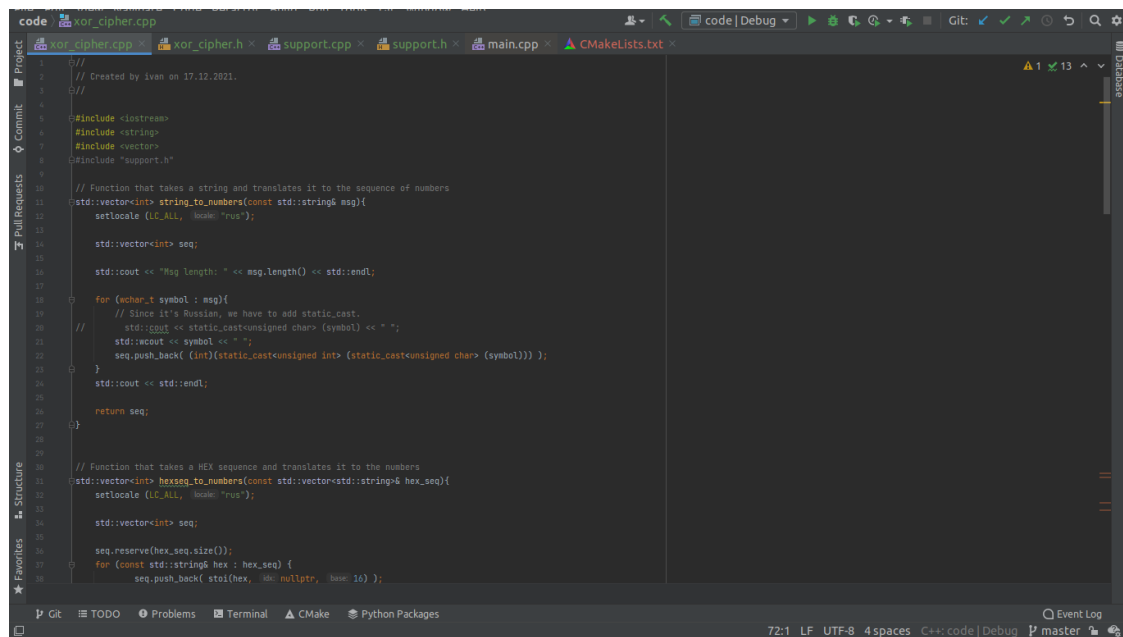
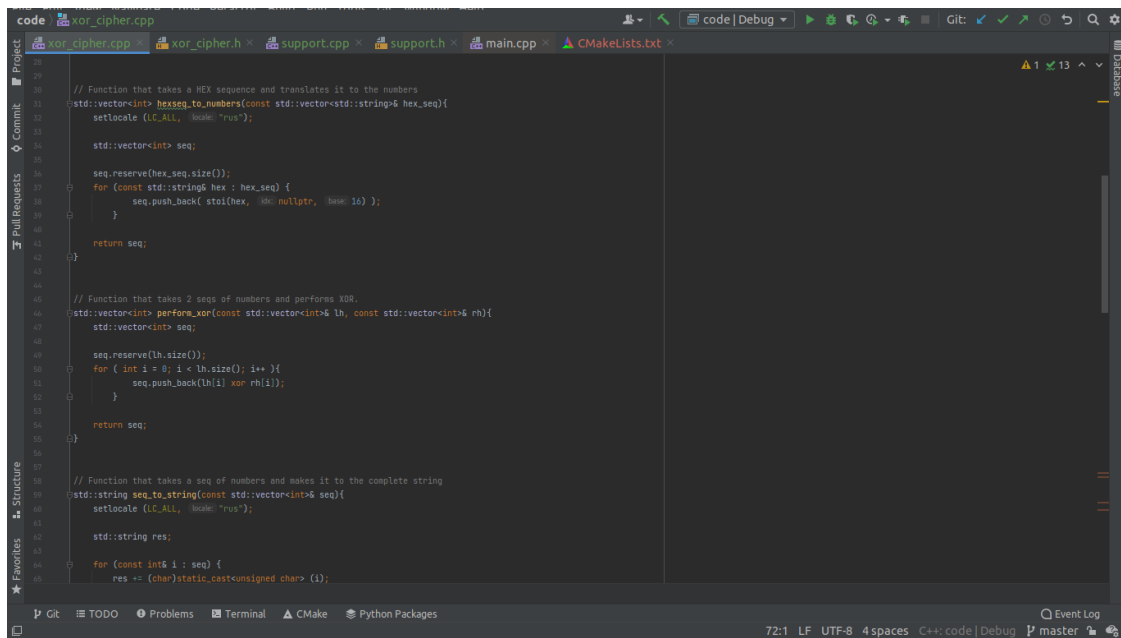


Figure 2: Kod 1



```
code xor_cipher.cpp
Project
Commit
Pull Requests
Structure
Favorites
// Function that takes a HEX sequence and translates it to the numbers
std::vector<int> hex_seq_to_numbers(const std::vector<std::string>& hex_seq){
    setlocale (LC_ALL, locale:: "rus");

    std::vector<int> seq;

    seq.reserve(hex_seq.size());
    for (const std::string& hex : hex_seq) {
        seq.push_back( stoi(hex, nullptr, 16) );
    }

    return seq;
}

// Function that takes 2 seq of numbers and performs XOR.
std::vector<int> perform_xor(const std::vector<int>& lh, const std::vector<int>& rh){
    std::vector<int> seq;

    seq.reserve(lh.size());
    for ( int i = 0; i < lh.size(); i++){
        seq.push_back(lh[i] xor rh[i]);
    }

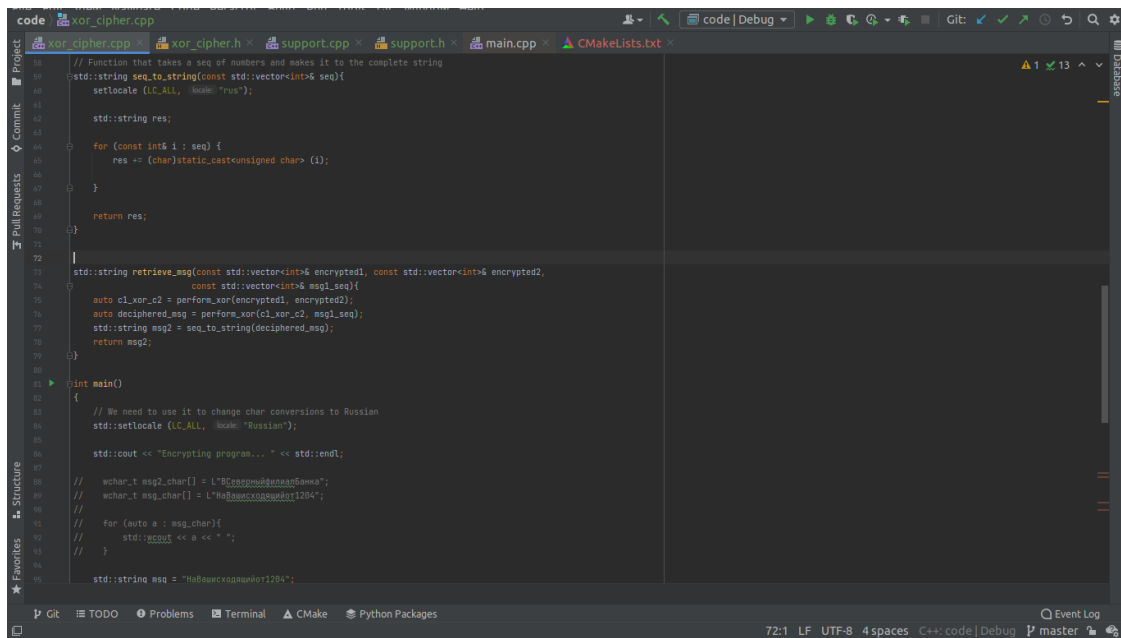
    return seq;
}

// Function that takes a seq of numbers and makes it to the complete string
std::string seq_to_string(const std::vector<int>& seq){
    setlocale (LC_ALL, locale:: "rus");

    std::string res;

    for (const int& i : seq) {
        res += (char)static_cast<unsigned char> (i);
    }
}
```

Figure 3: Kod 2



```
code xor_cipher.cpp
Project
Commit
Pull Requests
Structure
Favorites
// Function that takes a seq of numbers and makes it to the complete string
std::string seq_to_string(const std::vector<int>& seq){
    setlocale (LC_ALL, locale:: "rus");

    std::string res;

    for (const int& i : seq) {
        res += (char)static_cast<unsigned char> (i);
    }

    return res;
}

std::string retrieve_msg(const std::vector<int>& encrypted1, const std::vector<int>& encrypted2,
                        const std::vector<int>& msg1_seq){
    auto c1_xor_c2 = perform_xor(encrypted1, encrypted2);
    auto deciphered_msg = perform_xor(c1_xor_c2, msg1_seq);
    std::string msg2 = seq_to_string(deciphered_msg);
    return msg2;
}

int main()
{
    // We need to use it to change char conversions to Russian
    std::setlocale (LC_ALL, locale:: "Russian");

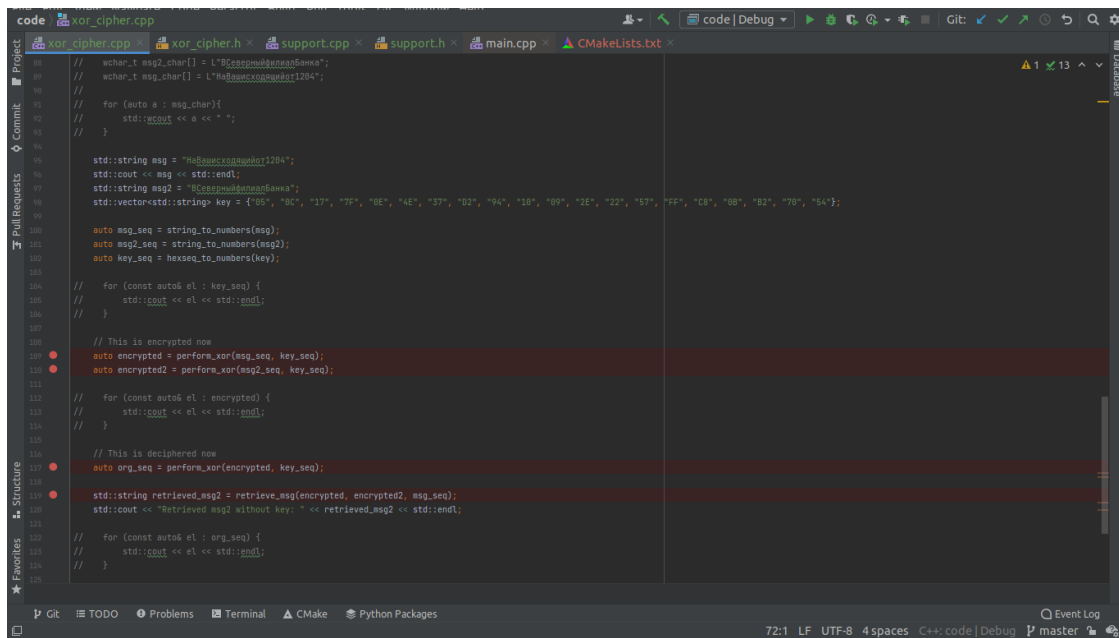
    std::cout << "Encrypting program... " << std::endl;

    wchar_t msg2_char[] = L"8C00p0u0@z0x050w0";
    wchar_t msg_char[] = L"Ha0aw0x0p0w00t01204";

    for (auto a : msg_char){
        std::wcout << a << " ";
    }

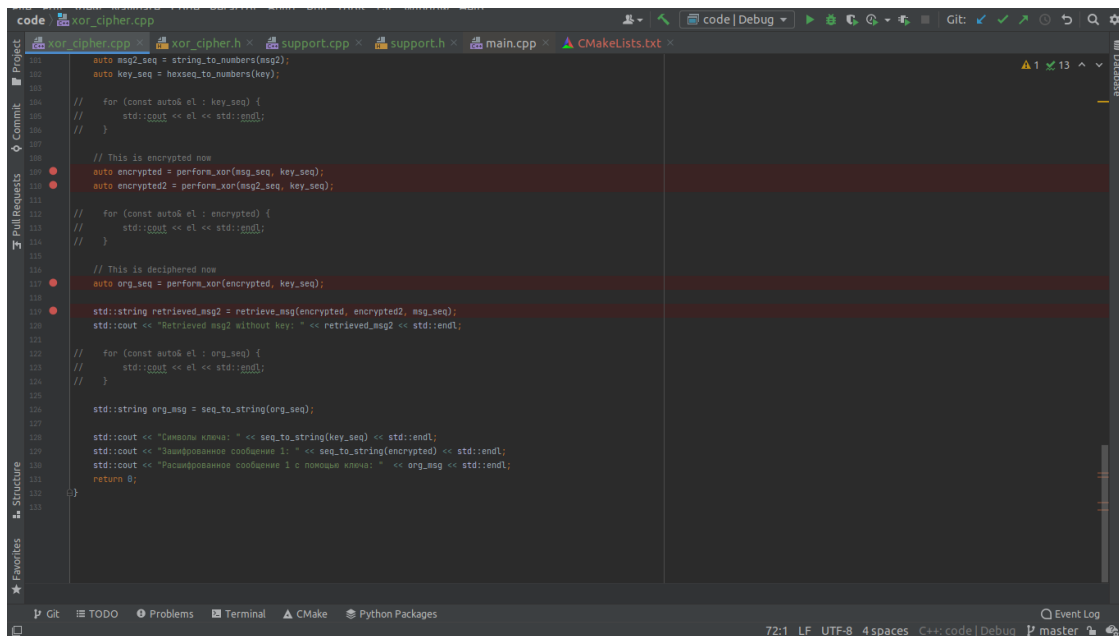
    std::string msg = "Ha0aw0x0p0w00t01204";
}
```

Figure 4: Kod 3



```
code | xor_cipher.cpp | xor_cipher.h | support.cpp | support.h | main.cpp | CMakeLists.txt
18 // wchar_t msg2_char[] = L"Восстановите сообщение";
19 // wchar_t msg_char[] = L"Восстановите сообщение";
20 //
21 // for (auto a : msg_char) {
22 //     std::cout << a << " ";
23 // }
24 //
25 std::string msg = "Восстановите сообщение";
26 std::cout << msg << std::endl;
27 std::string msg2 = "Восстановите сообщение";
28 std::vector<std::string> key = {"05", "06", "17", "7F", "0E", "4E", "37", "02", "94", "18", "09", "2E", "22", "57", "FF", "C8", "0B", "82", "70", "54"};
29
30 auto msg_seq = string_to_numbers(msg);
31 auto msg2_seq = string_to_numbers(msg2);
32 auto key_seq = hexseq_to_numbers(key);
33
34 // for (const auto& el : key_seq) {
35 //     std::cout << el << std::endl;
36 // }
37 //
38 // This is encrypted now
39 auto encrypted = perform_xor(msg_seq, key_seq);
40 auto encrypted2 = perform_xor(msg2_seq, key_seq);
41
42 // for (const auto& el : encrypted) {
43 //     std::cout << el << std::endl;
44 // }
45 //
46 // This is decrypted now
47 auto org_seq = perform_xor(encrypted, key_seq);
48
49 std::string retrieved_msg2 = retrieve_msg(encrypted, encrypted2, msg_seq);
50 std::cout << "Retrieved msg2 without key: " << retrieved_msg2 << std::endl;
51
52 // for (const auto& el : org_seq) {
53 //     std::cout << el << std::endl;
54 // }
55
56 Git | TODO | Problems | Terminal | CMake | Python Packages | Event Log
72:1 LF UTF-8 4 spaces C++ code | Debug | master
```

Figure 5: Код 4



```
code | xor_cipher.cpp | xor_cipher.h | support.cpp | support.h | main.cpp | CMakeLists.txt
181 auto msg2_seq = string_to_numbers(msg2);
182 auto key_seq = hexseq_to_numbers(key);
183
184 // for (const auto& el : key_seq) {
185 //     std::cout << el << std::endl;
186 // }
187 //
188 // This is encrypted now
189 auto encrypted = perform_xor(msg_seq, key_seq);
190 auto encrypted2 = perform_xor(msg2_seq, key_seq);
191
192 // for (const auto& el : encrypted) {
193 //     std::cout << el << std::endl;
194 // }
195 //
196 // This is decrypted now
197 auto org_seq = perform_xor(encrypted, key_seq);
198
199 std::string retrieved_msg2 = retrieve_msg(encrypted, encrypted2, msg_seq);
200 std::cout << "Retrieved msg2 without key: " << retrieved_msg2 << std::endl;
201
202 // for (const auto& el : org_seq) {
203 //     std::cout << el << std::endl;
204 // }
205
206 std::string org_msg = seq_to_string(org_seq);
207
208 std::cout << "Сообщение: " << seq_to_string(key_seq) << std::endl;
209 std::cout << "Зашифрованное сообщение 1: " << seq_to_string(encrypted) << std::endl;
210 std::cout << "Расшифрованное сообщение 1 с помощью ключа: " << org_msg << std::endl;
211 return 0;
212 }
213
214 Git | TODO | Problems | Terminal | CMake | Python Packages | Event Log
72:1 LF UTF-8 4 spaces C++ code | Debug | master
```

Figure 6: Код 5

4 Выводы

Освоил на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом