**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы программирования»

Отчет по лабораторной работе №8

«Шифрование текстовых файлов»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-15Б |  | преподаватель каф. ИУ5 |
| Коновалов Илья |  | Папшев И.С. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Задача:**

Разработать программу для кодирования и декодирования текста, хранящегося в файле, основанную на методе Цезаря, но более «изощренную»: ключ, используемый для кодирования символа, будет зависеть от позиции символа в исходном тексте. Для того, чтобы кодирование и декодирование можно было бы выполнять одним и тем же способом, вместо сложения кода символа с постоянным ключом (как у Цезаря) для переменных ключей используйте операцию «исключающее ИЛИ»: <код символа> ^ <ключ>.

Длина ключа – 1 байт.

Для получения ключей используется так называемый «кодовый блокнот», который представляет собой некоторый текст, известный только отправителю и получателю.

При кодировании с переменным ключом одинаковые символы исходного текста будут заменены в зашифрованном тексте различными символами, что существенно усложнит «взлом шифра». Количество символов в тексте не изменяется, а для кодирования и декодирования одинаковых по порядку символов в исходном и зашифрованном тексте должны использоваться одинаковые ключи.

Ключи представляют собой целые числа от 0 до 255, определяемые путем преобразования слов кодового блокнота по следующему алгоритму:

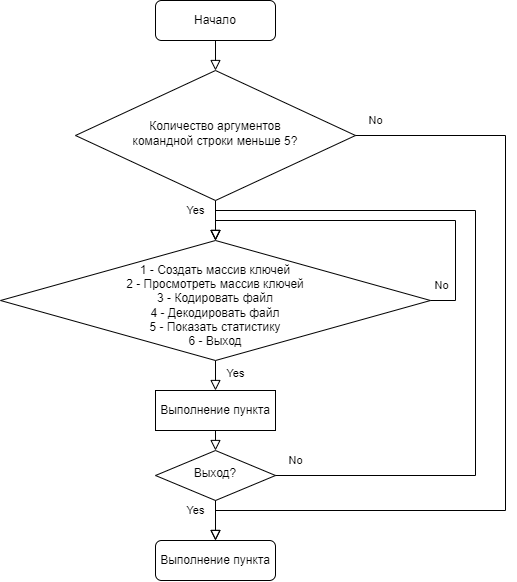
• Ключ для k-го по порядку символа в исходном тексте вычисляется как сумма (по модулю 256) кодов символов k-го по порядку слова кодового блокнота.

• Если кодовый блокнот имеет слов меньше, чем количество символов в исходном тексте, то по исчерпании слов в нём перейти к первому слову и продолжить кодирование.

Целесообразно сначала на основе кодового блокнота сформировать по заданному правилу целочисленный массив ключей, который затем будет использоваться при шифровании. Получение массива ключей оформить в виде отдельной функции и протестировать ее.

**Алгоритм:**

Блок-схема:



**Код программы:**

#include **<fstream>**#include **<iomanip>**#include **<iostream>  
  
unsigned char**\* create\_keys(**const char**\* code\_file\_name, size\_t length) {  
 **if** (!code\_file\_name) {  
 **return nullptr**;  
 }  
  
 std::ifstream code\_file(code\_file\_name, std::ios::in | std::ios::binary);  
  
 **if** (!code\_file) {  
 **return nullptr**;  
 }  
  
 **unsigned char**\* keys = **new unsigned char**[length];  
 **int** current\_key = 0;  
  
 **const int** max\_byte\_value = 256;  
  
 **for** (size\_t i = 0; i < length; ++i) {  
 **while** ((current\_key = code\_file.get()) != (**int**)**' '**) {  
 **if** (current\_key == EOF) {  
 code\_file.clear();  
 code\_file.seekg(0);  
 **break**;  
 }  
  
 keys[i] = (keys[i] + current\_key) % max\_byte\_value;  
 }  
 }  
  
 **return** keys;  
}  
  
**void** encode\_file(**const char**\* input\_file\_name, **const char**\* output\_file\_name, **const unsigned char**\* keys) {  
 **if** (!input\_file\_name || !output\_file\_name || !keys) {  
 **return**;  
 }  
  
 std::fstream input\_file(input\_file\_name, std::ios::in | std::ios::binary);  
  
 **if** (!input\_file) {  
 **return**;  
 }  
  
 std::fstream output\_file(output\_file\_name, std::ios::out | std::ios::binary);  
  
 **if** (!output\_file) {  
 **return**;  
 }  
  
 **int** current\_symbol = 0;  
 **for** (size\_t i = 0; (current\_symbol = input\_file.get()) != EOF; ++i) {  
 output\_file << (**unsigned char**)((**unsigned char**)current\_symbol ^ keys[i]);  
 }  
}  
  
**void** decode\_file(**const char**\* input\_file\_name, **const char**\* output\_file\_name, **const unsigned char**\* keys) {  
 encode\_file(input\_file\_name, output\_file\_name, keys);  
}  
  
**void** show\_keys(**const unsigned char**\* keys, size\_t length) {  
 **const int** num\_of\_byte\_digits = 3;  
 **const int** num\_of\_index\_digits = [length]() **mutable** {  
 **int** num\_of\_digits = 1;  
 **while** (length /= 10) {  
 ++num\_of\_digits;  
 }  
 **return** num\_of\_digits;  
 }();  
  
 **for** (size\_t i = 0; i < length; ++i) {  
 std::cout << **"key: (index) "** << std::setw(num\_of\_index\_digits) << i <<  
 **" (int value) "** << std::setw(num\_of\_byte\_digits) << (**int**)keys[i] << **'\n'**;  
 }  
}  
  
size\_t find\_file\_len(**const char**\* file\_name) {  
 **if** (!file\_name) {  
 **return** 0;  
 }  
  
 std::fstream file(file\_name, std::ios::in | std::ios::binary);  
  
 **if** (!file) {  
 **return** 0;  
 }  
  
 std::streampos file\_size = file.tellg();  
 file.seekg(0, std::ios::***end***);  
 file\_size = file.tellg() - file\_size;  
  
 **return** file\_size;  
}  
  
**void** find\_stat(**const char**\* encoded\_file\_name, **const char**\* input\_file\_name, **int**\* stat\_array, **char** sym) {  
 **if** (!encoded\_file\_name || !input\_file\_name || !stat\_array) {  
 **return**;  
 }  
  
 std::fstream encoded\_file(encoded\_file\_name, std::ios::in | std::ios::binary);  
 **if** (!encoded\_file) {  
 **return**;  
 }  
  
 std::fstream input\_file(input\_file\_name, std::ios::in | std::ios::binary);  
 **if** (!input\_file) {  
 **return**;  
 }  
  
 **int** current\_symbol = 0;  
 **for** (size\_t i = 0; (current\_symbol = input\_file.get()) != EOF; ++i) {  
 **if** (current\_symbol != (**int**)sym) {  
 **continue**;  
 }  
  
 stat\_array[encoded\_file.seekg(input\_file.tellg()).get()]++;  
 }  
}  
  
**int** main(**int** argc, **char**\*\* argv) {  
 **if** (argc < 5) {  
 std::cout << **"Недостаточно аргументов командной строки."** << std::endl;  
 **return** -1;  
 }  
  
 *// argv[1] - кодовый блокнот  
 // argv[2] - шифруемый файл  
 // argv[3] - файл, куда будет записан зашифрованный файл  
 // argv[4] - файл, куда будет записан дешифрованный файл* **const char**\* code\_note = argv[1];  
 **const char**\* file\_to\_encode = argv[2];  
 **const char**\* encoded\_file = argv[3];  
 **const char**\* decoded\_file = argv[4];  
  
 **const** size\_t num\_of\_keys = find\_file\_len(file\_to\_encode);  
  
 **char** option = 0;  
 **unsigned char**\* keys = **nullptr**;  
 **bool** is\_encoded = **false**;  
 **bool** is\_decoded = **false**;  
  
 **while** (option != **'q'**) {  
 std::cout << **"Команды:\n"** <<  
 **" 1 - Создать массив ключей\n"** <<  
 **" 2 - Просмотреть массив ключей\n"** <<  
 **" 3 - Кодировать файл\n"** <<  
 **" 4 - Декодировать файл\n"** <<  
 **" 5 - Показать статистику\n"** <<  
 **" q - Выйти"** << std::endl;  
  
 std::cin >> option;  
  
 **switch** (option) {  
 **case '1'**: {  
 **if** (!keys) {  
 keys = create\_keys(code\_note, num\_of\_keys);  
 }  
  
 std::cout << **"Success!"** << std::endl;  
 **break**;  
 }  
  
 **case '2'**: {  
 **if** (keys) {  
 show\_keys(keys, num\_of\_keys);  
 }  
 **break**;  
 }  
  
 **case '3'**: {  
 **if** (keys) {  
 **if** (!is\_encoded) {  
 encode\_file(file\_to\_encode, encoded\_file, keys);  
 is\_encoded = **true**;  
 }  
  
 std::cout << **"Success!"** << std::endl;  
 }  
 **break**;  
 }  
  
 **case '4'**: {  
 **if** (keys && is\_encoded) {  
 **if** (!is\_decoded) {  
 decode\_file(encoded\_file, decoded\_file, keys);  
 is\_decoded = **true**;  
 }  
  
 std::cout << **"Success!"** << std::endl;  
 }  
 **break**;  
 }  
  
 **case '5'**: {  
 **if** (is\_encoded) {  
 **int** stat[256] = {0};  
  
 **char** sym = 0;  
 std::cout << **"Введите символ для которого нужно показать статистику"** << std::endl;  
 std::cin >> sym;  
  
 find\_stat(encoded\_file, file\_to\_encode, stat, sym);  
  
 **for** (**int** i = 0; i < 16; ++i) {  
 **for** (**int** j = 0; j < 16; ++j) {  
 std::cout << std::setw(3) << stat[i \* 16 + j] << **' '**;  
 }  
  
 std::cout << **'\n'**;  
 }  
 }  
 }  
  
 **default**:  
 **continue**;  
 }  
 }  
  
 **delete**[] keys;  
  
 **return** 0;  
}

**Результаты:**

Таблица частотности пробельного символа при исходном тексте размером 76884 байта и с кодовым блокнотом размеров в 5120 байт:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 74 | 35 | 65 | 46 | 30 | 36 | 25 | 40 | 77 | 37 | 47 | 35 | 55 | 48 | 65 | 35 |
| 28 | 57 | 82 | 46 | 30 | 69 | 58 | 33 | 54 | 32 | 30 | 35 | 33 | 31 | 31 | 35 |
| 83 | 30 | 69 | 45 | 61 | 57 | 42 | 52 | 73 | 61 | 68 | 40 | 50 | 44 | 84 | 63 |
| 38 | 65 | 71 | 47 | 37 | 100 | 52 | 35 | 39 | 32 | 31 | 51 | 34 | 36 | 36 | 68 |
| 41 | 24 | 26 | 55 | 36 | 31 | 34 | 62 | 29 | 28 | 34 | 30 | 29 | 37 | 43 | 21 |
| 54 | 57 | 59 | 27 | 36 | 49 | 37 | 39 | 26 | 18 | 27 | 43 | 36 | 32 | 34 | 39 |
| 54 | 46 | 24 | 44 | 42 | 30 | 39 | 42 | 24 | 48 | 32 | 17 | 43 | 37 | 22 | 37 |
| 57 | 34 | 40 | 51 | 24 | 38 | 37 | 32 | 41 | 38 | 61 | 26 | 38 | 42 | 26 | 39 |
| 55 | 52 | 65 | 66 | 61 | 63 | 68 | 47 | 48 | 29 | 41 | 42 | 53 | 35 | 32 | 38 |
| 46 | 81 | 67 | 51 | 39 | 56 | 75 | 62 | 45 | 42 | 35 | 35 | 32 | 38 | 26 | 31 |
| 55 | 95 | 47 | 79 | 38 | 87 | 58 | 63 | 39 | 34 | 50 | 42 | 45 | 34 | 36 | 45 |
| 64 | 58 | 63 | 72 | 108 | 61 | 103 | 69 | 73 | 35 | 70 | 50 | 74 | 53 | 45 | 29 |
| 17 | 24 | 28 | 25 | 33 | 39 | 28 | 31 | 28 | 20 | 29 | 29 | 36 | 33 | 22 | 46 |
| 46 | 39 | 28 | 39 | 28 | 29 | 32 | 31 | 29 | 17 | 31 | 27 | 30 | 33 | 20 | 26 |
| 34 | 26 | 26 | 34 | 45 | 28 | 40 | 42 | 33 | 33 | 9 | 38 | 36 | 27 | 24 | 47 |
| 28 | 28 | 29 | 26 | 22 | 18 | 34 | 29 | 31 | 24 | 25 | 26 | 32 | 43 | 31 | 34 |

Таблица частотности пробельного символа при исходном тексте размеров 76884 байта и с кодовым блокнотом размером в 512 байт:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 71 | 40 | 77 | 35 | 50 | 33 | 21 | 42 | 65 | 74 | 65 | 54 | 65 | 74 | 67 | 46 |
| 34 | 64 | 99 | 43 | 31 | 100 | 65 | 33 | 52 | 33 | 29 | 49 | 30 | 42 | 26 | 53 |
| 106 | 45 | 61 | 79 | 53 | 58 | 30 | 64 | 68 | 96 | 70 | 43 | 63 | 81 | 88 | 64 |
| 44 | 85 | 77 | 37 | 37 | 114 | 79 | 35 | 49 | 35 | 27 | 37 | 45 | 38 | 26 | 61 |
| 43 | 45 | 38 | 68 | 36 | 53 | 41 | 78 | 15 | 38 | 47 | 39 | 38 | 40 | 35 | 24 |
| 93 | 44 | 87 | 37 | 53 | 54 | 42 | 49 | 26 | 25 | 44 | 65 | 73 | 29 | 50 | 52 |
| 51 | 44 | 41 | 57 | 24 | 56 | 33 | 54 | 23 | 47 | 29 | 24 | 42 | 44 | 32 | 18 |
| 102 | 48 | 58 | 51 | 52 | 39 | 51 | 64 | 46 | 55 | 51 | 51 | 40 | 42 | 36 | 46 |
| 56 | 58 | 53 | 31 | 43 | 22 | 119 | 56 | 25 | 20 | 42 | 19 | 28 | 25 | 24 | 21 |
| 38 | 38 | 43 | 57 | 35 | 63 | 67 | 55 | 28 | 27 | 31 | 26 | 41 | 44 | 16 | 24 |
| 88 | 142 | 50 | 44 | 42 | 77 | 51 | 39 | 23 | 12 | 27 | 29 | 34 | 15 | 16 | 22 |
| 53 | 62 | 39 | 46 | 85 | 63 | 69 | 76 | 40 | 31 | 74 | 35 | 68 | 58 | 10 | 14 |
| 24 | 37 | 20 | 18 | 19 | 39 | 29 | 13 | 34 | 12 | 8 | 28 | 9 | 29 | 11 | 43 |
| 51 | 14 | 26 | 31 | 34 | 29 | 10 | 33 | 24 | 28 | 26 | 16 | 35 | 27 | 27 | 6 |
| 13 | 33 | 23 | 14 | 50 | 45 | 26 | 41 | 19 | 18 | 8 | 17 | 29 | 21 | 7 | 33 |
| 23 | 10 | 34 | 36 | 24 | 18 | 10 | 23 | 19 | 21 | 45 | 5 | 17 | 36 | 31 | 14 |

Таблица частотности символа ‘a’ при исходном тексте размером 76884 байта и с кодовым блокнотом размеров в 5120 байт:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | 11 | 13 | 14 | 4 | 16 | 17 | 6 | 17 | 6 | 23 | 14 | 23 | 11 | 26 | 32 |
| 19 | 28 | 28 | 30 | 13 | 20 | 7 | 16 | 9 | 17 | 8 | 18 | 11 | 9 | 16 | 18 |
| 24 | 13 | 27 | 20 | 17 | 51 | 33 | 15 | 25 | 27 | 36 | 9 | 37 | 32 | 14 | 66 |
| 19 | 26 | 41 | 71 | 20 | 53 | 14 | 16 | 37 | 46 | 13 | 19 | 11 | 22 | 18 | 24 |
| 13 | 43 | 7 | 9 | 14 | 11 | 19 | 23 | 10 | 10 | 19 | 13 | 15 | 9 | 6 | 24 |
| 20 | 5 | 20 | 19 | 12 | 21 | 15 | 10 | 29 | 5 | 13 | 14 | 14 | 49 | 21 | 22 |
| 13 | 28 | 12 | 11 | 19 | 11 | 13 | 6 | 20 | 10 | 11 | 23 | 13 | 8 | 10 | 14 |
| 12 | 15 | 12 | 7 | 7 | 6 | 13 | 23 | 6 | 13 | 16 | 10 | 22 | 7 | 9 | 12 |
| 23 | 14 | 15 | 9 | 16 | 8 | 20 | 12 | 17 | 21 | 17 | 28 | 28 | 26 | 20 | 28 |
| 26 | 46 | 13 | 18 | 12 | 22 | 36 | 31 | 15 | 9 | 12 | 8 | 11 | 5 | 14 | 6 |
| 11 | 39 | 14 | 23 | 18 | 47 | 31 | 67 | 22 | 33 | 14 | 16 | 24 | 21 | 23 | 19 |
| 27 | 22 | 18 | 15 | 10 | 23 | 25 | 18 | 32 | 68 | 25 | 61 | 31 | 10 | 50 | 11 |
| 15 | 10 | 13 | 9 | 9 | 16 | 19 | 14 | 10 | 14 | 14 | 14 | 7 | 9 | 18 | 10 |
| 12 | 8 | 23 | 18 | 9 | 10 | 8 | 10 | 4 | 7 | 15 | 7 | 11 | 7 | 11 | 14 |
| 10 | 6 | 11 | 10 | 4 | 8 | 15 | 11 | 7 | 6 | 9 | 11 | 6 | 13 | 8 | 11 |
| 1 | 12 | 12 | 12 | 14 | 18 | 6 | 19 | 10 | 13 | 15 | 15 | 13 | 8 | 6 | 7 |

Таблица частотности символа ‘a’ при исходном тексте размером 76884 байта и с кодовым блокнотом размеров в 5120 байт:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 12 | 4 | 24 | 5 | 8 | 9 | 20 | 9 | 13 | 8 | 19 | 13 | 14 | 22 | 22 | 45 |
| 21 | 12 | 30 | 42 | 13 | 21 | 16 | 25 | 16 | 6 | 15 | 4 | 0 | 8 | 28 | 31 |
| 13 | 8 | 21 | 21 | 14 | 35 | 27 | 21 | 27 | 12 | 60 | 23 | 39 | 35 | 26 | 73 |
| 27 | 17 | 42 | 100 | 34 | 52 | 19 | 21 | 23 | 36 | 17 | 19 | 5 | 11 | 17 | 24 |
| 20 | 82 | 11 | 34 | 30 | 13 | 22 | 44 | 22 | 21 | 35 | 10 | 35 | 11 | 26 | 20 |
| 41 | 16 | 13 | 17 | 25 | 16 | 36 | 14 | 37 | 14 | 20 | 13 | 9 | 67 | 18 | 27 |
| 18 | 26 | 11 | 14 | 3 | 8 | 9 | 14 | 8 | 6 | 31 | 30 | 4 | 13 | 2 | 6 |
| 14 | 3 | 9 | 4 | 12 | 11 | 24 | 23 | 5 | 8 | 11 | 11 | 14 | 13 | 16 | 12 |
| 20 | 6 | 5 | 2 | 2 | 13 | 17 | 9 | 16 | 23 | 6 | 21 | 20 | 10 | 23 | 16 |
| 12 | 33 | 9 | 23 | 9 | 22 | 13 | 23 | 5 | 18 | 3 | 4 | 15 | 2 | 10 | 10 |
| 43 | 56 | 8 | 6 | 8 | 8 | 66 | 119 | 9 | 14 | 4 | 0 | 17 | 29 | 14 | 12 |
| 9 | 42 | 8 | 10 | 0 | 14 | 21 | 25 | 56 | 35 | 44 | 81 | 12 | 6 | 39 | 27 |
| 8 | 11 | 18 | 15 | 8 | 10 | 15 | 18 | 3 | 7 | 7 | 19 | 16 | 12 | 10 | 5 |
| 4 | 11 | 18 | 14 | 16 | 18 | 6 | 7 | 7 | 8 | 9 | 1 | 4 | 5 | 21 | 17 |
| 15 | 2 | 8 | 20 | 0 | 14 | 1 | 1 | 3 | 17 | 2 | 2 | 4 | 5 | 1 | 9 |
| 8 | 0 | 2 | 0 | 7 | 27 | 3 | 9 | 5 | 5 | 14 | 6 | 20 | 15 | 1 | 8 |

**Вывод:**

В ходе лабораторной работы научился работать с файлами и потоковым вводом/выводом. Разобрал алгоритм работы шифра Цезаря и реализовал его усовершенствованную версию с использованием кодового блокнота.