**Белорусский государственный технологический университет**

**Факультет информационных технологий**

**Кафедра программной инженерии**

**Программа “Шифрование файлов ”**

По дисциплине «Основы алгоритмизации и программирования»

Выполнил:

Студент 1 курса 10 группы ПИ

Макаревич Кирилл Витальевич

Проверил: Белодед Николай Иванович

Оглавление

[Введение 3](#_Toc181829200)

[Описание программы Coder.exe 4](#_Toc181829201)

[Описание программы Decoder.exe 7](#_Toc181829202)

[Работа программ 10](#_Toc181829203)

[Заключение 13](#_Toc181829204)

# **Введение**

Цель:

Разработать программы Coder.exe и Decoder.exe для шифрования и расшифрования текстовых файлов с использованием простого сдвига шифра. Работать с вводом\выводом программ с помощью командной строки.

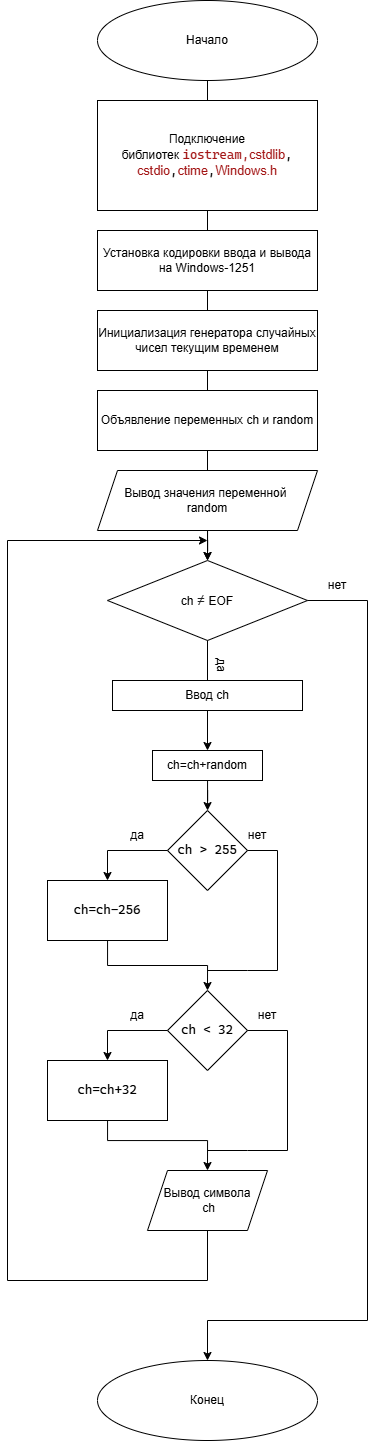
Задачи:

1. Создать программу Coder.exe для кодирования текста с добавлением случайного значения сдвига к каждому символу.
2. Создать программу Decoder.exe для восстановления исходного текста, используя обратный сдвиг на значение, которое использовалось при кодировании.

# **Описание программы Coder.exe**

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <cstdio>  #include <ctime>  #include <Windows.h>  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  srand((unsigned)time(NULL));  int ch, random = rand() % 10 + 1;  std::cout << random << std::endl; // вывод сдвига для дешифратора  while ((ch = getchar()) != EOF) {  ch = ch + random;  // ограничим результат в пределах от 32 до 255  if (ch > 255) {  ch -= 256;  }  if (ch < 32) {  ch += 32;  }  putchar(ch);  }  return 0;  } |

**Блок-схема:**



**Словесно-формульное описание алгоритма:**

1. Подключение библиотек
2. Установка кодировки
3. Инициализация генератора случайных чисел
4. Объявление переменных ch и random
5. Вывод значения переменной random
6. Цикл: Пока не достигнут конец файла (EOF)
7. Прибавление random к символу ch
8. Проверка: Если ch > 255, переход к шагу 9, иначе переход к шагу 10
9. Если ch > 255, уменьшаем его на 256
10. Проверка: Если ch < 32, переход к шагу 11, иначе переход к шагу 12
11. Если ch < 32, увеличиваем его на 32
12. Вывод символа ch, переход к шагу 6
13. Завершение программы

**Псевдоалгоритм:**

НАЧАЛО

СОЗДАТЬ переменные: ch, random

УСТАНОВИТЬ кодировку консоли:

ИНИЦИАЛИЗИРОВАТЬ генератор случайных чисел:

ВЫВЕСТИ значение переменной random

ПОКА ch ≠ EOF

НАЧАЛО ЦИКЛА

Ввод символа ch

ch=ch+random

ЕСЛИ ch > 255 ТО ch = ch - 256:

ИНАЧЕ переход к следующему шагу

ЕСЛИ ch < 32 ТО ch = ch + 32:

ИНАЧЕ переход к следующему шагу

ВЫВОД символа ch

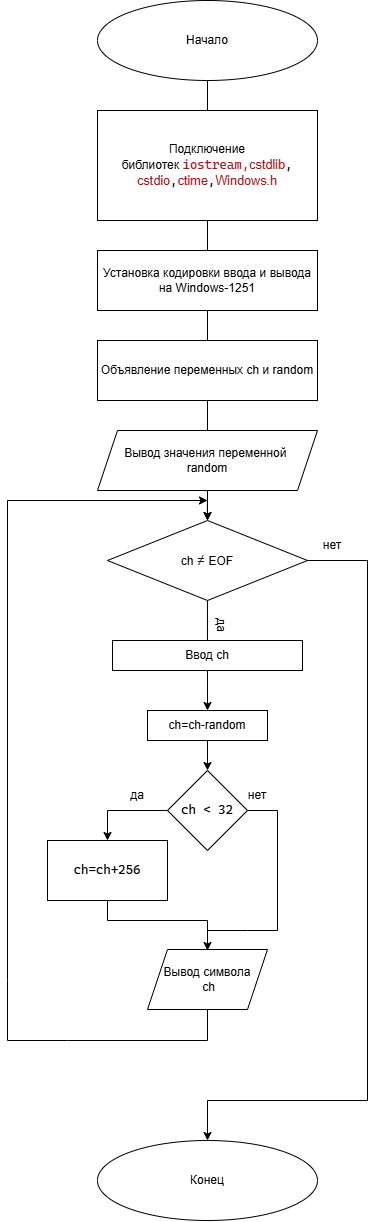
КОНЕЦ ЦИКЛА

КОНЕЦ

# **Описание программы Decoder.exe**

|  |
| --- |
| #include <iostream>  #include <cstdlib>  #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  int ch, random;  std::cin >> random; // ввод сдвига  while ((ch = getchar()) != EOF) {  ch = ch - random;  // ограничиваем результат в пределах от 32 до 255  if (ch < 32) {  ch += 256;  }  putchar(ch);  }  return 0;  } |

**Блок-схема:**



**Словесно-формульное описание алгоритма:**

1. Подключение библиотек
2. Установка кодировки
3. Объявление переменных ch и random
4. Вывод значения переменной random
5. Цикл: Пока не достигнут конец файла (EOF)
6. Вычитание random от символа ch
7. Проверка: Если ch < 32, переход к шагу 8, иначе переход к шагу 9
8. Если ch < 32, увеличиваем его на 256
9. Вывод символа ch, переход к шагу 5
10. Завершение программы

**Псевдоалгоритм:**

НАЧАЛО

СОЗДАТЬ переменные: ch, random

УСТАНОВИТЬ кодировку консоли:

ВЫВЕСТИ значение переменной random

ПОКА ch ≠ EOF

НАЧАЛО ЦИКЛА

Ввод символа ch

ch=ch-random

ЕСЛИ ch < 32 ТО ch = ch + 256:

ИНАЧЕ переход к следующему шагу

ВЫВОД символа ch

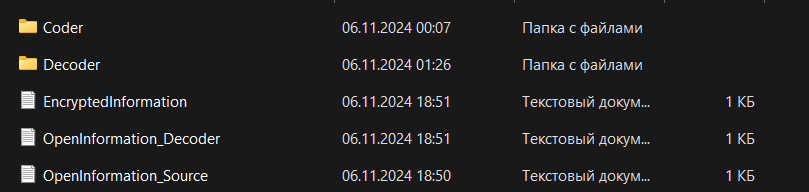
КОНЕЦ ЦИКЛА

КОНЕЦ

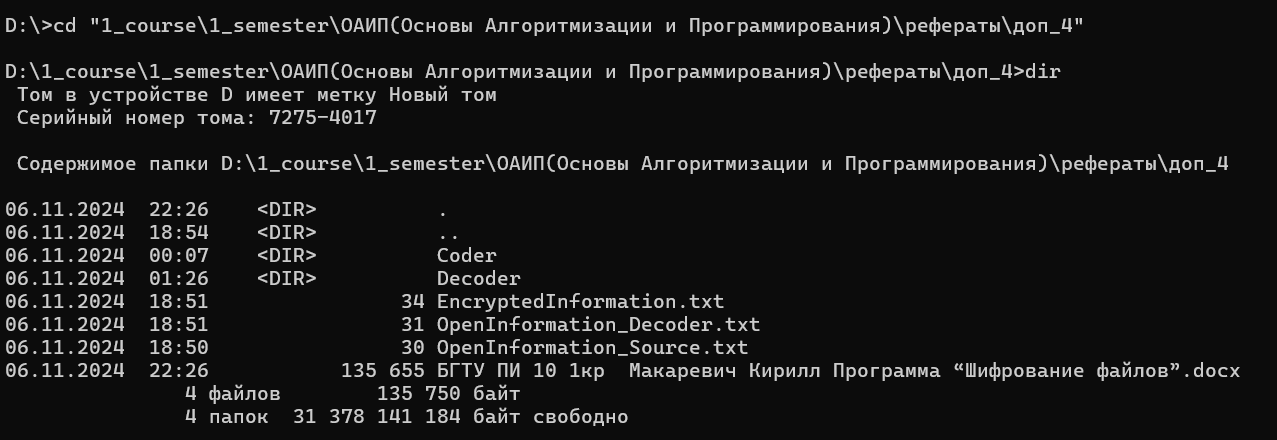
# **Работа программ**

**1. Структура моих документов:**

а) Скриншот в проводнике:

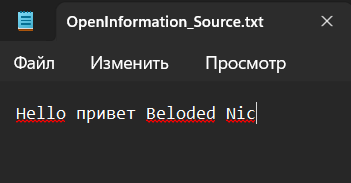
****

б) Скриншот в терминале:

****

**2. Содержимое файла OpenInformation\_Source.txt**

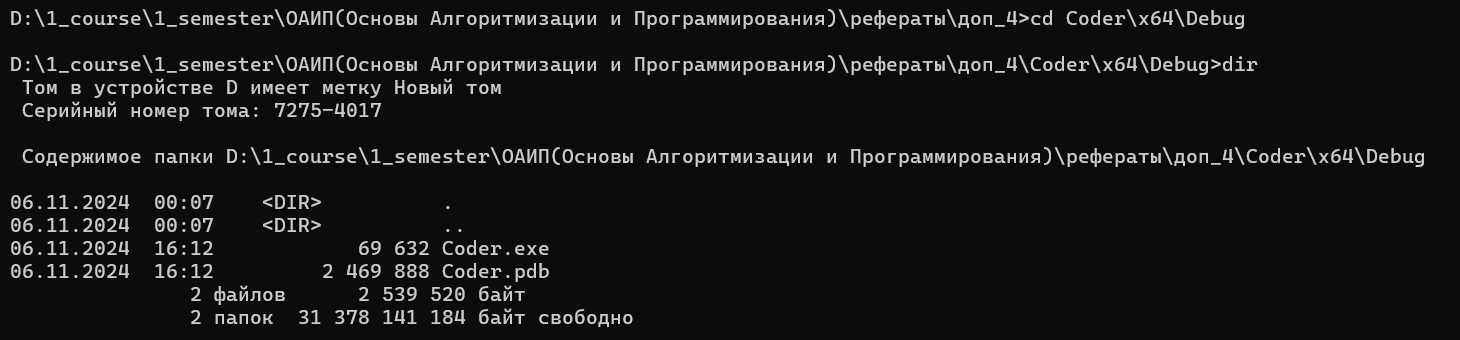
а) Скриншот в блокноте:

****

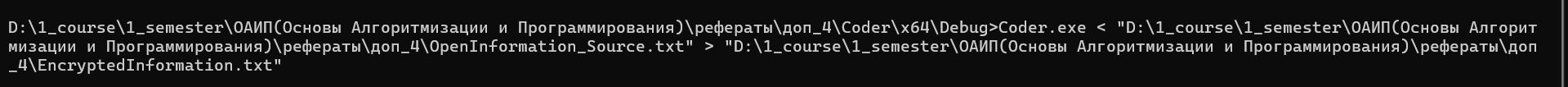
б) Скриншот в проводнике:

****

**3. Структура программы Coder**

****

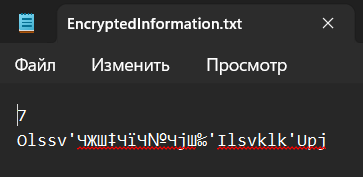
Запуск программы **Coder.exe (** читаем данные из файлаOpenInformation\_Source.txt и записываем в EncryptedInformation.txt )

****

|  |
| --- |
| **Текст команды:**  Coder.exe < "D:\1\_course\1\_semester\ОАИП(Основы Алгоритмизации и Программирования)\рефераты\доп\_4\OpenInformation\_Source.txt" > "D:\1\_course\1\_semester\ОАИП(Основы Алгоритмизации и Программирования)\рефераты\доп\_4\EncryptedInformation.txt" |

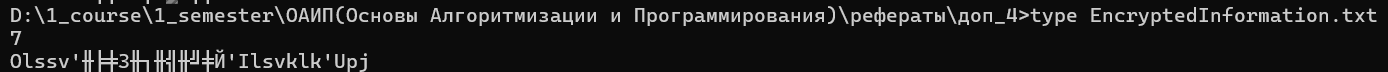
**4. Содержимое файла EncryptedInformation.txt**

а) Скриншот в блокноте:

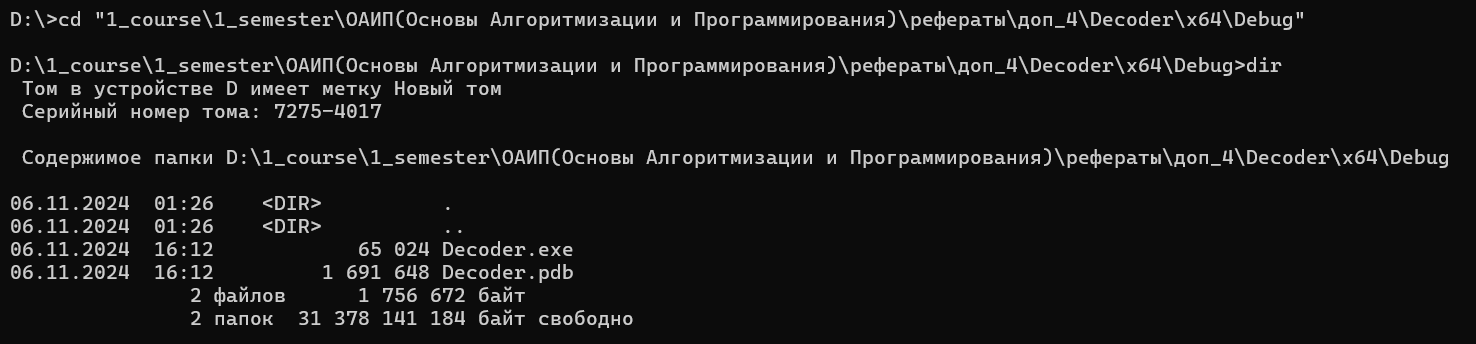
****

(на первой строке число 7, это значение переменной random, т.е. на сколько будет производится сдвиг, и эту переменную будет считывать Decoder.exe)

б) Скриншот в проводнике:

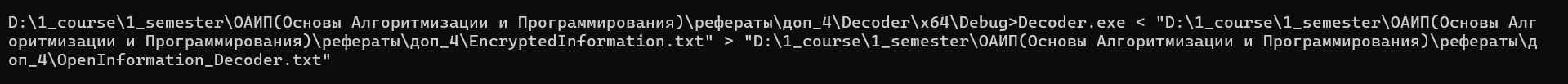
****

**5. Структура программы Decoder**

****

Запуск программы **Decoder.exe (** читаем данные из файлаEncryptedInformation.txt и записываем в

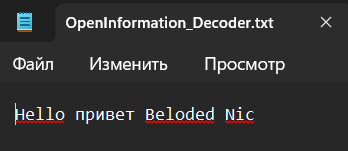
OpenInformation\_Decoder.txt)

****

|  |
| --- |
| **Текст команды:**  Decoder.exe < "D:\1\_course\1\_semester\ОАИП(Основы Алгоритмизации и Программирования)\рефераты\доп\_4\EncryptedInformation.txt" > "D:\1\_course\1\_semester\ОАИП(Основы Алгоритмизации и Программирования)\рефераты\доп\_4\OpenInformation\_Decoder.txt" |

**6. Содержимое файла OpenInformation\_Decoder.txt**

а) Скриншот в блокноте:

****

б) Скриншот в терминале

****

# **Заключение**

Разработанные программы `Coder.exe` и `Decoder.exe` успешно реализуют шифрование и дешифрование текстовых данных с использованием сдвига символов. Данный метод, несмотря на простоту, позволяет защитить текстовую информацию от случайного доступа и читается только при наличии значения сдвига, что может использоваться для базовой защиты данных. Этот проект показал, что даже простые алгоритмы могут быть полезны для задач начального уровня защиты данных, а также дал возможность освоить принципы работы с кодировками, файловыми потоками и базовыми механизмами шифрования в C++.