Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

09.03.01 Информатика и вычислительная техника, профиль: Программное обеспечение мобильных систем (очная форма обучения)

КУРСОВАЯ РАБОТА

Шифрование текста биграммами

Выполнил: студент гр. ИА-232 Московских Дмитрий Петрович

/

« » « »2023 г. (подпись)

Проверил: Преподаватель Лебеденко Людмила Федоровна

/

« » « » 2023 г. (подпись)

Новосибирск 2023 г.

Содержание

1. [Индивидуальное задание 2](#_TOC_250000)
   1. Тема курсовой работы [2](https://docs.google.com/document/d/1UpSVHRsDT1qcoZ4Wsi_7_AZnEH0dNrbdT52866D7-xo/edit#heading%3Dh.fupki7db20pp)
   2. Цель [2](https://docs.google.com/document/d/1UpSVHRsDT1qcoZ4Wsi_7_AZnEH0dNrbdT52866D7-xo/edit#heading%3Dh.4w78u6diofg5)
   3. Задание 2
   4. Шифр биграмм 2
2. Анализ задачи 5
   1. Алгоритм решения задачи 5
   2. Используемые функции 10
   3. Псевдокод 10
   4. Криптоанализ шифра биграмм 9
3. Тестовые данные 10
4. Итоги работы программы 11
5. Листинг программы 11
6. Заключение 16

Заключение

Список используемой литературы

# Индивидуальное задание

* 1. Тема

Шифрование текста биграммами

# Цель

Освоить методы шифрования – дешифрования текста.

# Задание

Разработать программу **Bigramm**, выполняющую шифрование в заданном тексте и **DeBigramm** – дешифровку текста. Текст до шифрования, после шифрования и после дешифровки должен выводиться на экран.

# Шифр биграмм

В начале XVI века аббат из Германии Иоганн Трисемус предложил шифровать по две буквы за раз. Шифры, использующие подобный принцип, были названы биграммными. Обычно такие шифры используют таблицы, аналогичные квадрату Полибия, заполненные символами используемого алфавита. Наиболее известный шифр биграммами называется Playfair. Он применялся Великобританией в Первую мировую войну. Открытый текст разбивался на пары букв (биграммы) и текст шифровки строился из него по следующим трём очень простым правилам.

1. Если обе буквы биграммы исходного текста принадлежат одной колонке таблицы, то буквами шифра считаются буквы, которые лежат под ними. Так би- грамма ИН дает текст шифровки НЗ. Если буква открытого текста находится в нижнем ряду, то для шифра берется соответствующая буква из верхнего ряда и биграмма НЯ дает шифр ЗИ. (Биграмма из одной буквы или пары одинаковых букв тоже подчиняется этому правилу, и текст ОО дает шифр ГГ).

2 Если обе буквы биграммы исходного текста принадлежат одной строке таблицы, то буквами шифра считаются буквы, которые лежат справа от них. Так биграмма АБ дает текст шифровки НГ. Если буква открытого текста находится в крайней правой колонке, то для шифра берется буква из крайней левой колонки той же строки и биграмма АД дает шифр НА.

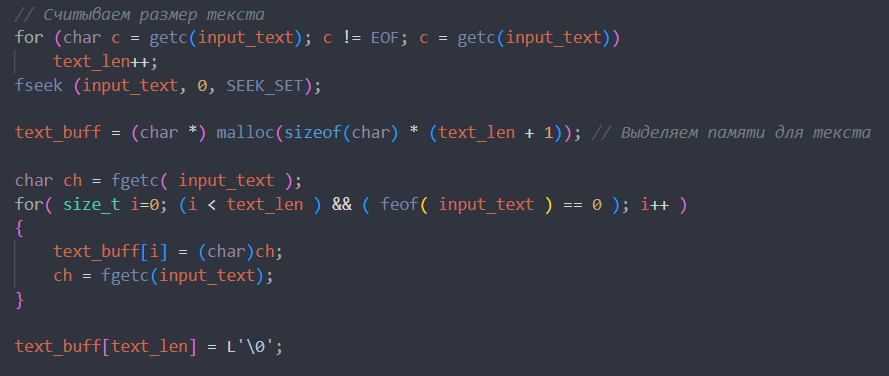
3. Если обе буквы биграммы открытого текста лежат в разных рядах и колонках, то вместо них берутся такие две буквы, чтобы вся их четверка представляла прямоугольник. Например, биграмма ЕК шифруется как БЙ (КЕ зашифруется ЙБ).

Заполнение квадрата алфавитом может быть случайным, а может определяться некоторой ключевой фразой, все символы которой (но без повторений) записываются в начале матрицы, а затем по порядку выписываются остальные буквы алфавита.

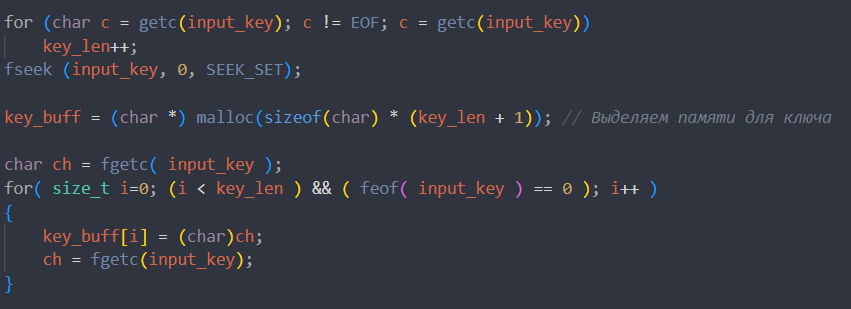
# Анализ задачи

# Алгоритм решения задачи

1. Проверяем количество элементов в заданном тексте динамического выделения памяти:



1. Подсчитываем и выделяем память для хранения ключа :



1. Передаём ключ(или NULL) в функцию шифрования/дешифрования: Изображение выглядит как текст, Шрифт, снимок экрана

   Автоматически созданное описание
2. Выделяем памяти для результата шифрования и дешифрования, создаём словарную матрицу:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

4. Принцип создания матрицы словаря с ключом

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

6. Вычисляем позиции символов в матрице

Изображение выглядит как текст, Шрифт, снимок экрана, типография

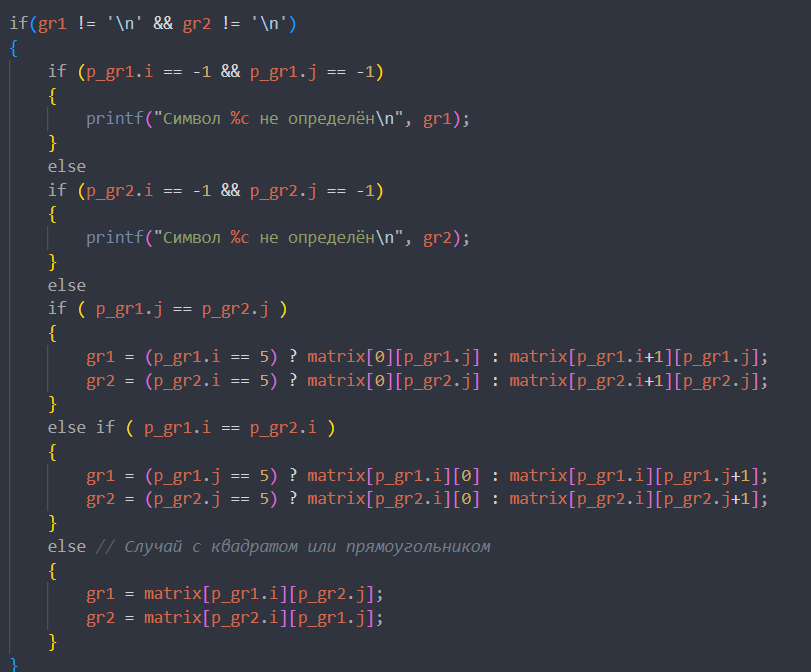
Автоматически созданное описание

, где p\_gr1, p\_gr2 имеет следующую структуру

Изображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Автоматически созданное описание

1. Шифруем согласно правилам
2. Изображение выглядит как текст, снимок экрана, дисплей

   Автоматически созданное описаниеДешифруем текст Изображение выглядит как текст, снимок экрана, дисплей

   Автоматически созданное описание
3. Записываем результаты в фаил, выводим на экран 
4. Сравниваем дешифрованный вариант с исходным текстом. Изображение выглядит как текст, снимок экрана, Шрифт

   Автоматически созданное описание

где bool CryptoCompare(size\_t len\_compare); Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Очистка матрицы словаря:

Изображение выглядит как текст, Шрифт, снимок экрана, рукописный текст

Автоматически созданное описание

1. Закрытие потока исходных файлов  
   Изображение выглядит как Шрифт, текст, Графика, типография

   Автоматически созданное описание
   1. Используемые функции

В данном коде используются следующие встроенные функции:

* + 1. fopen(): Функция открывает файл и возвращает указатель на FILE-объект, который используется для доступа к файлу. В коде используется для открытия файлов "text.txt", "key.txt", "decode.txt" и "encode.txt" в разных режимах доступа (чтение и запись)..
    2. fgetc(): Функция считывает символ из файла.
    3. strlen(): Функция возвращает длину строки, используется в циклах.
    4. printf(): Функция форматированного вывода строки в файл.
    5. fclose(): Функция закрывает файл, предварительно сбрасывая все буферы. Принимает указатель на файл, который нужно закрыть.
    6. setlocale(): Функция устанавливает текущую локаль программы в зависимости от настроек операционной системы. В данном случае, пустая строка "Rus" передана в качестве аргумента, чтобы использовать русскую локализацию.
    7. malloc(): Функция выделяет блок памяти указанного размера в байтах. Она принимает размер в байтах в качестве аргумента и возвращает указатель на выделенный блок памяти.
    8. free(): Функция освобождает блок памяти, который был ранее выделен с помощью функции malloc(). Она принимает указатель на блок памяти, который нужно освободить.
    9. strcat(): На вход функции подаётся две строки, одна исходная, другая та, которую нужно склеить с исходной, возвращается конкакетация двух строк
  1. Псевдокод

Установить язык(русский)

Создать указатель input\_text = NULL, input\_key = NULL;

If(аргумент1 существует)

{

input\_text = открыть фаил(аргумент1, читать)

}

If(аргумент2 существует)

{

Input\_key = открыть фаил(аргумент1, читать)

}

Создать указатель text = NULL;

Text = выделить память(размер текста\*вес одного символа в байтах);

If(аргумент1 существует)

Создать указатель key = NULL

Key = выделить память(размер ключа\*вес одного символа в байтах);

CryptoIO(передать: текст, длину текста, ключ, длина ключа)

* CryptoIO

выделить память для результата(размер текста\*вес символа в байтах)

выделить память для результата2(размер текста\*вес символа в байтах)

for(size\_t CryptoID равен 0, меньше длины текста - 1, увеличить на 2)

{

Создать символ gr1, gr2

Поместить в gr1: text[CryptoID]

Поместить в gr2: text[CryptoID+1]

If(символ не ‘\n’)

{

If(Символ1 не существует в словаре)

{

Символ не найден

}

Else

If(Символ2 не существует в словаре)

{

Символ2 не найден

}

Else

If(Символ1 и Символ2 в одном столбце

{

Gr1 = если в конце строки, то символ ставим на начало, иначе меняем символ на другой строке

Gr2 - аналогично

}

Else

If(Символы на одной строке)

{

Gr1 = если символ в конце столбца, то вернём в начало, иначе перместим в следующий столец

}

Else

{

Gr1 ставим позицию символа Gr2

Gr2 ставим позицию символа Gr1

}

}

Дешифровка(обратная операция)

Ввести текст на экран

Вывести текст в фаил

* 1. Криптоанализ

При дешифровании текста используют частотные характеристики открытого текста. Однако для получения устойчивой картины длина послания должна быть существенно больше ключа. Одной из наиболее устойчивых характеристик осмысленного текста является отсутствие запретных биграмм (пара соседних букв). Алгоритм не является криптоустойчивым т.к. существует алгоритм, который может раскрыть послание за время меньшее, чем через перебор

# Тестовые данные

Мы заполняем 2 файла: text.txt – вводится слово которое требуется зашифровать на русском языке, key.txt – фаил, содержащий ключ на русском языке (опционально)

* + 1. Осуществялется сборка файла, путём запуска make.bat

Код: gcc -shared -Os -s -o crypto.dll lib/crypto.c

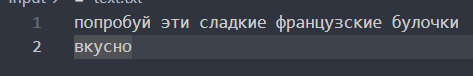
gcc -Os -s bigramm.c crypto.dll -o crypto\_start

rm \*.o

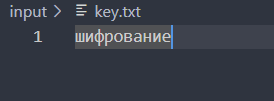
* + 1. Ввод команды запуска: ./crypto\_start.exe путь\_до\_файла\_с\_текстом путь\_до\_файла\_с\_ключём(опционально)

рассмотрим запуск с ключём:

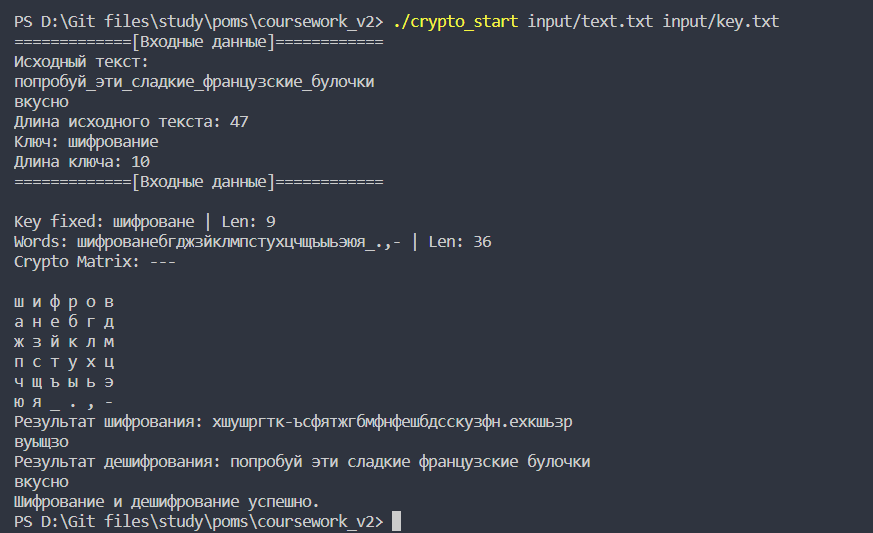
Исходный текст:



Ключ:



ВАЖНО: использовать текст, написанный маленькими буквами

Запись происходит в файлы output/decode.txt и output/encode.txt

Использование программы без ключа:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

# Итоги работы программы

Программа:

* + - * Считывает начальное слово.
      * Создаёт матрицу словаря.
      * Шифрует слово и записывает его в файл.
      * Дешифрует слово и проверяет его на сходство и изначальным словом.
      * Выводит начальное слово, зашифрованное и дешифрованное.

# Листинг программы

1. Crypto.h:

#ifndef CRYPTO\_LIB\_H

#define CRYPTO\_LIB\_H

#include "stdio.h"

#include "stdlib.h"

#include "stdbool.h"

#include "string.h"

#include "malloc.h"

*////////////////////*

void CryptoIO(char \*text, size\_t text\_len, char \*key, size\_t key\_len);

bool CryptCompare(size\_t text\_len);

#endif

1. Crypto.c
2. #include "crypto.h"
3. char alphabet[] = "абвгдежзийклмнопрстуфхцчшщъыьэюя\_.,-";
4. const int alphabet\_size = 37;
5. typedef struct Pos {
6. int i;
7. int j;
8. } POS;
9. void PrintMatrix(char \*\*m, size\_t size)
10. {
11. printf("\n");
12. for(size\_t row = 0; row < size; row++)
13. {
14. for(size\_t col = 0; col < size; col++)
15. {
16. printf("%c ", m[row][col]);
17. }
18. printf("\n");
19. }
20. }
21. void RemoveDuplicates(char \*str, size\_t key\_len)
22. {
23. int len = key\_len;
24. int index = 0;
25. for (int i = 0; i < len; i++) {
26. int j;
27. for (j = 0; j < i; j++) {
28. if (str[i] == str[j]) {
29. break;
30. }
31. }
32. if (j == i) {
33. str[index++] = str[i];
34. }
35. }
36. str[index] = L'\0';
37. }
39. POS GetCharPos(char c, char \*\*matrix)
40. {
41. POS pos;
42. pos.i = -1;
43. pos.j = -1;
44. for(size\_t row = 0; row < 6; row++)
45. {
46. for(size\_t col = 0; col < 6; col++)
47. {
48. if(matrix[row][col] == c)
49. {
50. pos.i = row;
51. pos.j = col;
52. }
53. }
54. }
55. return pos;
56. }
57. char \*\*CryptoMatrix(char \*key, size\_t key\_len)
58. {
59. *//Удаление из ключа лишних букв*
60. char \*words\_array;
61. if (key != NULL)
62. {
63. RemoveDuplicates(key, key\_len); *// удаляем*
64. printf("Key fixed: %s | Len: %d\n", key, strlen(key));
65. size\_t len\_matrix = strlen(key) + alphabet\_size + 1;
66. words\_array = (char \*)malloc(sizeof(char) \* len\_matrix);
67. for(size\_t i = 0; i < key\_len; i++)
68. words\_array[i] = key[i];
69. strcat(words\_array, alphabet);
70. RemoveDuplicates(words\_array, len\_matrix);
71. printf("Words: %s | Len: %d\n", words\_array, strlen(words\_array));
72. }
73. else
74. {
75. size\_t len\_matrix = alphabet\_size + 1;
76. words\_array = (char \*)malloc(sizeof(char) \* len\_matrix);
77. for(size\_t i = 0; i < alphabet\_size; i++)
78. {
79. words\_array[i] = alphabet[i];
80. }
81. words\_array[alphabet\_size] = '\0';
83. printf("Words: %s | Len: %d\n", words\_array, strlen(words\_array));
84. }
85. char \*\*cryptoMatrix = (char \*\*)malloc(6\*sizeof(char \*));
86. for(int i = 0; i < 6; i++)
87. cryptoMatrix[i] = (char \*)malloc(6\*sizeof(char));
88. size\_t wordidx = 0;
89. for(size\_t row = 0; row < 6; row++)
90. {
91. for(size\_t col = 0; col < 6; col++)
92. {
93. cryptoMatrix[row][col] = words\_array[wordidx];
94. wordidx++;
95. }
96. }
97. return cryptoMatrix;
98. }
99. void CryptoIO(char \*text, size\_t text\_len, char \*key, size\_t key\_len)
100. {
101. char \*result\_e = (char \*)malloc(sizeof(char) \* text\_len);
102. char \*result\_d = (char \*)malloc(sizeof(char) \* text\_len);
103. char \*\*matrix = CryptoMatrix(key, key\_len);
105. printf("Crypto Matrix: ---\n");
106. PrintMatrix(matrix, 6);
107. for(size\_t CryptoID = 0; CryptoID < text\_len - 1; CryptoID += 2)
108. {
109. char gr1, gr2;
110. gr1 = text[CryptoID];
111. gr2 = text[CryptoID+1];
112. POS p\_gr1, p\_gr2;
113. p\_gr1 = GetCharPos(gr1, matrix);
114. p\_gr2 = GetCharPos(gr2, matrix);
115. if(gr1 != '\n' && gr2 != '\n')
116. {
117. if (p\_gr1.i == -1 && p\_gr1.j == -1)
118. {
119. printf("Символ %c не определён\n", gr1);
120. }
121. else
122. if (p\_gr2.i == -1 && p\_gr2.j == -1)
123. {
124. printf("Символ %c не определён\n", gr2);
125. }
126. else
127. if ( p\_gr1.j == p\_gr2.j )
128. {
129. gr1 = (p\_gr1.i == 5) ? matrix[0][p\_gr1.j] : matrix[p\_gr1.i+1][p\_gr1.j];
130. gr2 = (p\_gr2.i == 5) ? matrix[0][p\_gr2.j] : matrix[p\_gr2.i+1][p\_gr2.j];
131. }
132. else if ( p\_gr1.i == p\_gr2.i )
133. {
134. gr1 = (p\_gr1.j == 5) ? matrix[p\_gr1.i][0] : matrix[p\_gr1.i][p\_gr1.j+1];
135. gr2 = (p\_gr2.j == 5) ? matrix[p\_gr2.i][0] : matrix[p\_gr2.i][p\_gr2.j+1];
136. }
137. else *// Случай с квадратом или прямоугольником*
138. {
139. gr1 = matrix[p\_gr1.i][p\_gr2.j];
140. gr2 = matrix[p\_gr2.i][p\_gr1.j];
141. }
142. }
143. result\_e[CryptoID] = gr1;
144. result\_e[CryptoID + 1] = gr2;
145. }
146. if (text\_len % 2 != 0)
147. {
148. result\_e[text\_len-1] = text[text\_len-1];
149. }
150. for(size\_t CryptoID = 0; CryptoID < text\_len - 1; CryptoID += 2)
151. {
152. char gr1, gr2;
153. gr1 = result\_e[CryptoID];
154. gr2 = result\_e[CryptoID+1];
155. POS p\_gr1, p\_gr2;
156. p\_gr1 = GetCharPos(gr1, matrix);
157. p\_gr2 = GetCharPos(gr2, matrix);
158. if(gr1 != '\n' && gr2 != '\n')
159. {
160. if (p\_gr1.i == -1 && p\_gr1.j == -1)
161. {
162. printf("Символ %c не определён\n", gr1);
163. }
164. else
165. if (p\_gr2.i == -1 && p\_gr2.j == -1)
166. {
167. printf("Символ %c не определён\n", gr2);
168. }
169. else
170. if ( p\_gr1.j == p\_gr2.j )
171. {
172. gr1 = (p\_gr1.i == 0) ? matrix[5][p\_gr1.j] : matrix[p\_gr1.i-1][p\_gr1.j];
173. gr2 = (p\_gr2.i == 0) ? matrix[5][p\_gr2.j] : matrix[p\_gr2.i-1][p\_gr2.j];
174. }
175. else if ( p\_gr1.i == p\_gr2.i )
176. {
177. gr1 = (p\_gr1.j == 0) ? matrix[p\_gr1.i][5] : matrix[p\_gr1.i][p\_gr1.j-1];
178. gr2 = (p\_gr2.j == 0) ? matrix[p\_gr2.i][5] : matrix[p\_gr2.i][p\_gr2.j-1];
179. }
180. else *// случай с квадратом или прямоугольником*
181. {
182. gr1 = matrix[p\_gr1.i][p\_gr2.j];
183. gr2 = matrix[p\_gr2.i][p\_gr1.j];
184. }
185. }
187. result\_d[CryptoID] = gr1;
188. result\_d[CryptoID + 1] = gr2;
189. }
190. if (text\_len % 2 != 0)
191. {
192. result\_d[text\_len-1] = result\_e[text\_len-1];
193. }
194. for(size\_t i = 0; i < text\_len; i++)
195. {
196. if(result\_d[i] == '\_')
197. {
198. result\_d[i] = ' ';
199. }
200. }
202. for(size\_t i = 0; i < 6; i++)
203. {
204. free(matrix[i]);
205. }
206. free(matrix);
207. FILE \*encode\_result = fopen("out/encode.txt", "w");
208. FILE \*decode\_result = fopen("out/decode.txt", "w");
209. if(!encode\_result || !decode\_result)
210. {
211. printf("Ошибка открытия файла\n");
212. exit(1);
213. }
214. printf("Результат шифрования: ");
215. for(size\_t i = 0; i < text\_len; i++)
216. {
217. printf("%c", result\_e[i]);
218. fputc(result\_e[i], encode\_result);
219. }
220. printf("\n");
221. printf("Результат дешифрования: ");
222. for(size\_t i = 0; i < text\_len; i++)
223. {
224. printf("%c", result\_d[i]);
225. fputc(result\_d[i], decode\_result);
226. }
227. printf("\n");
228. fclose(encode\_result);
229. fclose(decode\_result);
231. free(result\_e);
232. free(result\_d);
233. }
234. bool CryptCompare(size\_t text\_len)
235. {
236. FILE \*source = fopen("input/text.txt", "r");
237. FILE \*dest = fopen("out/decode.txt", "r");
238. bool Compare = true;
239. for(size\_t i = 0; i < text\_len; i++)
240. {
241. if (fgetc(source) != fgetc(dest))
242. Compare = false;
243. }
244. fclose(source);
245. fclose(dest);
246. return Compare;
247. }
248. Bigramm.c
249. #include "lib/crypto.h"
250. #include "locale.h"
251. int main(int argc, char \*argv[])
252. {
253. setlocale(LC\_ALL, "Rus");
254. FILE \*input\_text = NULL, \*input\_key = NULL;
255. if (argv[1] != NULL)
256. {
257. input\_text = fopen(argv[1], "r");
258. }
259. if(argv[2] != NULL) {
260. input\_key = fopen(argv[2], "r");
261. }
262. if ( (argv[1] != NULL && !input\_text) || (argv[2] != NULL && !input\_key ) )
263. {
264. printf("\nНевозможно открыть файлы.\n");
265. exit(0);
266. }
268. if ( argv[1] == NULL && argv[2] == NULL )
269. {
270. printf("Ничего не делать.\n");
271. exit(0);
272. }
273. char \*key\_buff = NULL;
274. char \*text\_buff = NULL;
275. size\_t key\_len = 0;
276. size\_t text\_len = 0;
277. *// Считываем размер текста*
278. for (char c = getc(input\_text); c != EOF; c = getc(input\_text))
279. text\_len++;
280. fseek (input\_text, 0, SEEK\_SET);
281. text\_buff = (char \*) malloc(sizeof(char) \* (text\_len + 1)); *// Выделяем памяти для текста*
282. char ch = fgetc( input\_text );
283. for( size\_t i=0; (i < text\_len ) && ( feof( input\_text ) == 0 ); i++ )
284. {
285. text\_buff[i] = (char)ch;
286. ch = fgetc(input\_text);
287. }
288. text\_buff[text\_len] = L'\0';
290. size\_t index = 0;
291. while(text\_buff[index] != '\0')
292. {
293. if(text\_buff[index] == ' ')
294. {
295. text\_buff[index] = '\_';
296. }
297. index++;
298. }
299. printf("=============[Входные данные]============\n");
300. printf("Исходный текст: \n%s\nДлина исходного текста: %d\n", text\_buff, strlen(text\_buff));
301. if(argv[2] != NULL)
302. {
303. for (char c = getc(input\_key); c != EOF; c = getc(input\_key))
304. key\_len++;
305. fseek (input\_key, 0, SEEK\_SET);
306. key\_buff = (char \*) malloc(sizeof(char) \* (key\_len + 1)); *// Выделяем памяти для ключа*
307. char ch = fgetc( input\_key );
308. for( size\_t i=0; (i < key\_len ) && ( feof( input\_key ) == 0 ); i++ )
309. {
310. key\_buff[i] = (char)ch;
311. ch = fgetc(input\_key);
312. }
313. key\_buff[key\_len] = '\0';
314. printf("Ключ: %s\nДлина ключа: %d\n", key\_buff, key\_len);
315. printf("=============[Входные данные]============\n\n");
316. CryptoIO(text\_buff, text\_len, key\_buff, key\_len);
317. if (CryptCompare(text\_len))
318. {
319. printf("Шифрование и дешифрование успешно.\n");
320. }
321. else
322. {
323. printf("Шифрование и дешифрование не успешно.\n");
324. }
325. }
326. else
327. {
328. CryptoIO(text\_buff, text\_len, NULL, 0);
329. if (CryptCompare(text\_len))
330. {
331. printf("Шифрование и дешифрование успешно.\n");
332. }
333. else
334. {
335. printf("Шифрование и дешифрование не успешно.\n");
336. }
337. }
338. fclose(input\_text);
339. fclose(input\_key);
340. return 0;
341. }
342. Заключение

В данной курсовой работе был представлен код на языке программирования C для выполнения операций кодирования и декодирования текста с использованием шифра биграмм.

Код включает несколько функций, каждая из которых отвечает за определенную задачу. Функция CryptoIO осуществляет кодирование и декодирование текста.

Код также содержит основную функцию main(int args, char \*argc[]), которая является точкой входа в программу. В этой функции происходит открытие и закрытие файлов для чтения и записи данных, кодирования и декодирования текста. Программа также проверяет открытия файла и выводит ошибку при не корректном открытие файла.

В заключении можно отметить, что код успешно выполняет задачи кодирования и декодирования текста с использованием шифра биграмм, а выполнение данной курсовой работы позволило углубить знания в области программирования на языке C.

Данный код так же выложен на платформе github для всеобщего анализа и дополнение к портфолио.

Ссылка на директорию:

<https://github.com/DmitriyMosk/study/tree/main/poms/coursework_v2>

Список используемой литературы

1. Руководство"C Programming" от Wikibooks [Электронный ресурс] <https://en.wikibooks.org/wiki/C_Programming>
2. Документация Microsoft Developer Network (MSDN) для функций и библиотек языка C. [Электронный ресурс]

<https://learn.microsoft.com/ru-ru/cpp/c-language/?view=msvc-170>

1. Интернет-сайт Stack Overflow [Электронный ресурс] <https://stackoverflow.com/>
2. Интернет-сайт GitHub [Электронный ресурс] <https://stackoverflow.com/>
3. Интернет-сайт Crypto101 по алгоритмам шифрования [Электронный ресурс]

<https://www.crypto101.io/>

1. Онлайн-книга "Learn C" от программиста "cprogramming.com" [Электронный ресурс]

<https://www.learn-c.org/>