1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и технологий
5. **Кафедра «Информационная безопасность компьютерных систем»**

**Лабораторная работа 8**

1. «Основы криптоанализа»
2. по дисциплине «Практикум по информационной безопасности»

Выполнил:

студент гр. 13558/1 Никулкин В. А.

Проверил:

ассистент преподавателя Алексеев. И. В.

1. Санкт-Петербург

2017

1. **Цель работы**

Приобретение навыков криптоанализа, ознакомление с дешифрованием криптограмм на примере частотного метода.

1. **Ход работы**
2. Была написана программа, осуществляющая следующие функции

* Анализ частоты букв входного файла и вывод предполагаемых замен в соответствии с частотами распределения букв русского алфавита
* Вывод на экран всех слов, сгруппированных по количеству букв
* Вывод на экран всех слов, сгруппированных по количеству нерасшифрованных на данный момент букв
* Отображение расшифрованного на данный момент текста построчно.
* Возможность замены букв
* Возможность отмены последней замены букв
* Интерфейс взаимодействия с пользователем
* Вывод на экран всех выполненных на данный момент замен

1. Листинг программы

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <locale.h>

#include <string.h>

struct text{

unsigned char \*initial;

unsigned char \*replacements;

};

void print\_text(struct text a)

{

printf("%s\n", a.initial);

}

void print\_lines(struct text a, int n)

{

int i = 0;

int counter = 0;

while(1)

{

if((a.initial[i] == '\0') || (a.initial[i] == '\n'))

++counter;

if(counter == n)

break;

++i;

}

unsigned char \*s = (unsigned char\*)calloc(i, sizeof(unsigned char));

int r = 0;

printf("\n");

while(r < i)

{

s[r] = a.initial[r];

++r;

}

s[r] = '\0';

printf("%s", s);

free(s);

}

void zamena(struct text c, unsigned char a, unsigned char b)

{

int i;

int k = 0;

while(c.initial[k] != '\0')

++k;

char\* s = (char\*)malloc(k \* sizeof(char));

for(int j = 0; j < k; ++j)

s[j] = c.initial[j];

for(i = 0; i < k; ++i)

{

if(c.initial[i] == a)

c.initial[i] = b;

}

int l = a - 192;

c.replacements[l] = b;

c.initial[0] = s[0];

for(int j = 1; j < k; ++j)

{

if(s[j] == '.')

{

++j;

while(s[j] == ' ')

++j;

c.initial[j] = s[j];

}

}

printf("%s\n", c.initial);

free(s);

}

void cancel(struct text a, unsigned char b)

{

int l;

int i;

int j;

int k = 0;

while(a.initial[k] != '\0')

++k;

for(i = 0; i < k; ++i)

{

if(a.initial[i] == b)

{

for(j = 0; j < 32; ++j)

{

if(a.replacements[j] == b)

break;

}

a.initial[i] = 192 + j;

l = a.initial[i] - 192;

}

}

a.replacements[l] = '-';

printf("%s\n", a.initial);

}

void show\_replacements(struct text a)

{

int i;

unsigned char c = 192;

int counter = 1;

for(i = 0; i < 32; ++i)

{

printf("%d) %c -> %c\n", counter, c, a.replacements[i]);

++c;

++counter;

}

}

void sort\_length(struct text a, int n)

{

int k = 0;

int j;

int i = 0;

int counter = -1;

while(a.initial[i] != '\0')

{

++counter;

if((a.initial[i] == ' ') || (a.initial[i] == '\n'))

{

if(counter == n)

{

j = 0;

char \*s = (char\*)malloc(n \* sizeof(char));

k = 0;

for(j = i - n; j < i; ++j)

{

s[k] = a.initial[j];

++k;

}

s[k] = '\0';

printf("%s\n", s);

free(s);

}

counter = -1;

}

++i;

}

}

void sort\_undeciphered(struct text a, int n)

{

int k = 0;

int j;

int i = 0;

int counter = -1;

int flag = 0;

int counter1 = 0;

while(a.initial[i] != '\0')

{

++counter;

if((a.initial[i] == ' ') || (a.initial[i] == '\n'))

{

j = 0;

unsigned char \*s = (unsigned char\*)calloc(counter, sizeof(unsigned char));

k = 0;

for(j = i - counter; j < i; ++j)

{

s[k] = a.initial[j];

++k;

}

s[k] = '\0';

for(flag = 0; flag < k; ++flag)

{

if((s[flag] >= 192) && (s[flag] <= 223))

counter1++;

}

if(counter1 == n)

printf("%s\n", s);

counter = -1;

counter1 = 0;

free(s);

}

++i;

}

}

void chastota(struct text a)

{

unsigned char au[32];

au[0] = 238; au[1] = 229; au[2] = 224; au[3] = 232; au[4] = 242; au[5] = 237; au[6] = 241; au[7] = 240; au[8] = 226; au[9] = 235;

au[10] = 234; au[11] = 236; au[12] = 228; au[13] = 239; au[14] = 243; au[15] = 255 ; au[16] = 251; au[17] = 252; au[18] = 227;

au[19] = 231; au[20] = 225; au[21] = 247; au[22] = 233; au[23] = 245; au[24] = 230; au[25] = 248; au[26] = 254; au[27] = 246;

au[28] = 249; au[29] = 253; au[30] = 244; au[31] = 250;

int j;

int o = 0;

int counter = 0;

int i = 0;

int k = 0;

unsigned char c = 192;

double buf[33];

unsigned char s[33];

float res;

while(a.initial[k] != '\0')

++k;

int u = k;

while(c <= 223)

{

for(i = 0; i < k; ++i)

{

if(a.initial[i] == c)

++counter;

}

if(counter == 0)

buf[c - 192] = 0;

else

{

res = (float)counter / (float)k;

buf[c - 192] = res;

}

++c;

counter = 0;

}

int l = 0;

int index = 0;

double max = buf[0];

while(o < 32)

{

for(l = 0; l < 32; ++l)

{

if(buf[l] > max)

{

max = buf[l];

index = l;

}

}

s[index] = au[o];

//printf("%lf\n", max);

buf[index] = -10;

++o;

max = -1;

}

unsigned char symbol = 192;

int counter1 = 1;

for(int u = 0; u < 32; ++u)

{

printf("%d) %c -> %c\n", counter1, symbol, s[u]);

++symbol;

++counter1;

}

}

void auto\_zamena(struct text a)

{

unsigned char au[32];

au[0] = 238; au[1] = 229; au[2] = 224; au[3] = 232; au[4] = 242; au[5] = 237; au[6] = 241; au[7] = 240; au[8] = 226; au[9] = 235;

au[10] = 234; au[11] = 236; au[12] = 228; au[13] = 239; au[14] = 243; au[15] = 255 ; au[16] = 251; au[17] = 252; au[18] = 227;

au[19] = 231; au[20] = 225; au[21] = 247; au[22] = 233; au[23] = 245; au[24] = 230; au[25] = 248; au[26] = 254; au[27] = 246;

au[28] = 249; au[29] = 253; au[30] = 244; au[31] = 250;

int j;

int o = 0;

int counter = 0;

int i = 0;

int k = 0;

unsigned char c = 192;

double buf[33];

unsigned char s[33];

float res;

while(a.initial[k] != '\0')

++k;

int u = k;

while(c <= 223)

{

for(i = 0; i < k; ++i)

{

if(a.initial[i] == c)

++counter;

}

if(counter == 0)

buf[c - 192] = 0;

else

{

res = (float)counter / (float)k;

buf[c - 192] = res;

}

++c;

counter = 0;

}

int l = 0;

int index = 0;

double max = buf[0];

while(o < 32)

{

for(l = 0; l < 32; ++l)

{

if(buf[l] > max)

{

max = buf[l];

index = l;

}

}

s[index] = au[o];

buf[index] = (float)-1;

++o;

max = -1;

}

unsigned char symbol = 192;

int counter1 = 0;

for(int u = 0; u < 32; ++u)

a.replacements[u] = s[u];

while(counter1 < 32)

{

for(l = 0; l < k; ++l)

{

if(a.initial[l] == symbol)

a.initial[l] = s[counter1];

}

++counter1;

++symbol;

}

a.initial[k] = '\0';

printf("%s\n", a.initial);

}

void conclusion(struct text a)

{

FILE \*out = fopen("out.txt", "w");

fprintf(out, "%s", a.initial);

fclose(out);

free(a.initial);

free(a.replacements);

}

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

struct text a;

int i;

a.replacements = (unsigned char\*) malloc(sizeof(unsigned char) \* 32);

for(i = 0; i < 32; ++i)

a.replacements[i] = '-';

FILE \*in = fopen("8.txt", "r");

fseek(in, 0, SEEK\_END);

int length = ftell(in);

fseek(in, 0, SEEK\_SET);

a.initial = (unsigned char\*) malloc(sizeof(unsigned char) \* length);

fread (a.initial, 1, length, in);

a.initial[length] = '\0';

int o;

printf("Справка по использованию:\n");

printf("1 - Замена буквы\n2 - Отмена замены буквы\n3 - вывести текст\n4 - вывести определенное количество строк теста\n5 - сортировка слов по длине\n");

printf("6 - сортировка слов по количеству нерасшифрованных букв\n7 - вывод предполагаемых замен по частному распределению\n8 - автозамена по частному распределению\n");

printf("9 - советы по расшифровке\n10 - ключ шифрования\n11 - завершить работу\n12 - справка по использованию\n");

unsigned char c;

int k;

unsigned char b;

while(1)

{

fflush(stdin);

printf("Что будем делать?\n");

scanf("%d", &k);

if(k == 1)

{

printf("Введите заменяемую букву\n");

fflush(stdin);

scanf("%c", &c);

fflush(stdin);

printf("Введите предполагаемую замену\n");

scanf("%c", &b);

zamena(a, c, b);

continue;

}

if(k == 2)

{

fflush(stdin);

printf("Введите какую замену отменить\n");

scanf("%c", &c);

cancel(a, c);

fflush(stdin);

continue;

}

if(k == 3)

{

print\_text(a);

continue;

}

if(k == 4)

{

fflush(stdin);

int n;

printf("Введите сколько строк текста ввыести\n");

scanf("%d", &n);

print\_lines(a, n);

fflush(stdin);

continue;

}

if(k == 5)

{

fflush(stdin);

int n;

printf("Введите длину сортируемых слов\n");

scanf("%d", &n);

sort\_length(a, n);

fflush(stdin);

continue;

}

if(k == 6)

{

fflush(stdin);

int n;

printf("Введите количество нерасшифрованных ьукв в слове\n");

scanf("%d", &n);

sort\_undeciphered(a, n);

fflush(stdin);

continue;

}

if(k == 7)

{

fflush(stdin);

chastota(a);

continue;

}

if(k == 8)

{

fflush(stdin);

auto\_zamena(a);

continue;

}

if(k == 9)

{

printf("\n1 - если есть тире, и после него млово из трех букв, вероятно это слово это\n2 - если есть множество повторяющихся слов из одной буквы, то это скорее всего союз и\n");

continue;

}

if(k == 10)

{

show\_replacements(a);

continue;

}

if(k == 11)

{

fflush(stdin);

conclusion(a);

return 0;

}

if(k == 12)

{

fflush(stdin);

printf("Справка по использованию:\n");

printf("1 - Замена буквы\n2 - Отмена замены буквы\n3 - вывести текст\n4 - вывести определенное количество строк теста\n5 - сортировка слов по длине\n");

printf("6 - сортировка слов по количеству нерасшифрованных букв\n7 - вывод предполагаемых замен по частному распределению\n8 - автозамена по частному распределению\n");

printf("9 - советы по расшифровке\n10 - ключ шифрования\n11 - завершить работу\n12 - справка по использованию\n");

continue;

}

}

}

1. Была расшифрована криптограмма

КФТЦЯЕИЛДУЮП АОДЧР - ХИФ ЦДФЙДЖТТЖ, МФИФДЖЬ ТФЗЛИ НЖДЖЗЖИЯ КДЧЙОЛ ЦДФЙДЖТТЮ, ТФКОЩОШОДЧЬ ОЫ ЦФРДЛКРИАФТ

КФГЖАСЛУОЬ РАФЛП, АФНТФЗУФ ОНТЛУЛУУФП, МФЦОО. ЗЖДЖЗЖЬ ЦДФЙДЖТТЮ, АОДЧР ТФЗЛИ ДЖРЦДФРИДЖУЬИЯРЬ ЦФ МФТЦЯЕИЛДУФП

РОРИЛТЛ ОСО РЛИО, ОРЦФСЯНЧЬ ЦФСУФТФБОЬ ЦФСЯНФАЖИЛСЛП КСЬ НЖДЖЗЛУОЬ ОЫ ЦДФЙДЖТТ. КЖЗКЖЬ ЦДФЙДЖТТЖ, НЖДЖЗЛУУЖЬ

АОДЧРФТ, ИЖМЗЛ КЛПРИАЧЛИ МЖМ АОДЧР, БИФ ФЦДЛКЛСЬЛИ ДЖРЦДФРИДЖУЛУОЛ ОУЩЛМШОО А МФТЦЯЕИЛДУФП РОРИЛТЛ ОСО РЛИО.

СОЙУЖИЧДЖ АОДЧРЖ - ФЦДЛКЛСЛУУЖЬ ЦФРСЛКФАЖИЛСЯУФРИЯ ГЖПИ, МФИФДЖЬ АРИДЛБЖЛИРЬ ИФСЯМФ А АОДЧРЖЫ О ГФСЯЪЛ УОЙКЛ.

СОЙУЖИЧДФП ТФЗЛИ ЬАСЬИЯРЬ ИЛМРИФАЖЬ РИДФМЖ, ФЦДЛКЛСЛУУЖЬ ЦФРСЛКФАЖИЛСЯУФРИЯ МФТЖУК, АЮЦФСУЬЛТЮЫ АОДЧРФТ О И.Ц.

ПФОРМ ИЖМОЫ ЧБЖРИМФА ЬАСЬЛИРЬ ФКУОТ ОН РЦФРФГФА ФГУЖДЧЗЛУОЬ АОДЧРФА, ЫФИЬ ИЖМОЛ ЧБЖРИМО ТФЙЧИ О УЛ РФКЛДЗЖИЯРЬ А

ИЛСЛ АОДЧРЖ. ПДОТЛДФТ ИЖМОЫ АОДЧРФА ЬАСЬЕИРЬ ЦФСОТФДЩУЮЛ АОДЧРЮ, МФИФДЮЛ ОНТЛУЬЕИ РАФП МФК ЦДО МЖЗКФТ ЦФРСЛКЧЕВЛТ

НЖДЖЗЛУОО.

ПФТОТФ АОДЧРФА РДЛКО ДЖНДЧЪЖЕВОЫ ЦДФЙДЖТТУЮЫ РДЛКРИА (РПС) РЧВЛРИАЧЕИ ИЖМ УЖНЮАЖЛТЮЛ "ИДФЬУРМОЛ МФУО" - ЦДФЙДЖТТЮ,

МФИФДЮЛ РФКЛДЗЖИ РМДЮИЮП ТФКЧСЯ, ФРЧВЛРИАСЬЕВОП УЛРЖУМШОФУОДФАЖУУЮЛ КЛПРИАОЬ (УЖЦДОТЛД, БИЛУОЛ МФУЩОКЛУШОЖСЯУЮЫ

КЖУУЮЫ). ТЖМОЛ ЦДФЙДЖТТЮ УЛ ЬАСЬЕИРЬ АОДЧРЖТО, ИЖМ МЖМ УЛ ОТЛЕИ РЦФРФГУФРИО М ДЖНТУФЗЛУОЕ. РЖНУФАОКУФРИЯЕ

ИДФЬУРМОЫ ЦДФЙДЖТТ ЬАСЬЕИРЬ СФЙОБЛРМОЛ ГФТГЮ, МФИФДЮЛ АЮЦФСУЬЕИ УЛРЖУМШОФУОДФАЖУУЮЛ КЛПРИАОЬ СОЪЯ ЦДО

УЖРИЧЦСЛУОО ФЦДЛКЛСЛУУЮЫ ЧРСФАОП (УЖЦДОТЛД, УЖРИЧЦСЛУОЛ МЖМФП-СОГФ КЖИЮ), Ж КФ ХИФЙФ ТФТЛУИЖ АЛКЧИ РЛГЬ

МЖМ ФГЮБУЮЛ ЦДФЙДЖТТЮ, УЛ РФАЛДЪЖЕВОЛ УОМЖМОЫ АДЛКУЮЫ КЛПРИАОП.

ПДО АЮЦФСУЛУОО НЖДЖЗЛУУФП ЦДФЙДЖТТЮ ЧЦДЖАСЛУОЛ ЦЛДЛКЖЛИРЬ АОДЧРЧ, О ФУ НЖДЖЗЖЛИ КДЧЙОЛ ЦДФЙДЖТТЮ О АЮЦФСУЬЛИ

КЛРИДЧМИОАУЮЛ КЛПРИАОЬ, Ж ЦФИФТ ЦЛДЛКЖЛИ ЧЦДЖАСЛУОЛ ЦДФЙДЖТТЛ-УФРОИЛСЕ. ПФХИФТЧ АУЛЪУЛ, МЖМ ЦДЖАОСФ,

НЖДЖЗЛУУЖЬ ЦДФЙДЖТТЖ АЛКЛИ МЖМ УЛНЖДЖЗЛУУЖЬ.

.

Предположим, что слово ХИФ после тир это ЭТО, заменим буквы.

поэтоТЧ скорее всего ПОЭТОМУ, сделаем замену.

Цотом это вероятно ПОТОМ, заменим Ц на П.

Есть много повторяющихся букв О, предположим, что это И.

пДи вероятно ПРИ, заменим Д на Р.

Предположим моЗЛт это может. Заменим буквы.

иЫ скорее всего ИХ. Заменим Ы на Х.

Вероятно Мопии это КОПИИ. Заменим М на К.

Рети вероятно СЕТИ, заменим Р на С.

Аирус скорее всего ВИРУС, заменим А на В.

компЯЕтерУЮП вирус, можно сделать вывод, что это слово КОМПЬЮТЕРНЫЙ.

проЙрЖммЖ это вероятно ПРОГРАММА, сделаем соответствующие замены.

котораЬ это КОТОРАЯ, заменим Ь на Я.

Наражать, вероятно заражать. Заменим Н на З.

Кругие вероятно ДРУГИЕ, сделаем замену.

Системе иСи сети, вероятно С надо заменить на Л.

доГавления это добавления, Заменим Г на Б.

внеЪне вероятно ВНЕШНЕ, заменим Ъ на Ш.

разрушаюВих это РАЗРУШАЮЩИХ, заменим В на Щ.

несанкШионированные это НЕСАНКЦИОНИРОВАННЫЕ, заменим Ш на Ц.

модиЩицирую это МОДИФИЦИРУЮ, заменим Щ на Ф.

полномоБия это вероятно ПОЛНОМОЧИЯ, заменим Б на Ч.

Получается, Э соответствует Ъ, так как все остальные соответствия установлены.

Получаем такую криптограмму:

компьютерный вирус - это программа, которая может заражать другие программы, модифицируя их посредством

добавления своей, возможно измененной, копии. заражая программы, вирус может распространяться по компьютерной

системе или сети, используя полномочия пользователей для заражения их программ. каждая программа, зараженная

вирусом, также действует как вирус, что определяет распространение инфекции в компьютерной системе или сети.

сигнатура вируса - определенная последовательность байт, которая встречается только в вирусах и больше нигде.

сигнатурой может являться текстовая строка, определенная последовательность команд, выполняемых вирусом и т.п.

поиск таких участков является одним из способов обнаружения вирусов, хотя такие участки могут и не содержаться в

теле вируса. примером таких вирусов являются полиморфные вирусы, которые изменяют свой код при каждом последующем

заражении.

помимо вирусов среди разрушающих программных средств (сйл) существуют так называемые "троянские кони" - программы,

которые содержат скрытый модуль, осуществляющий несанкционированные действия (например, чтение конфиденциальных

данных). такие программы не являются вирусами, так как не имеют способности к размножению. разновидностью

троянских программ являются логические бомбы, которые выполняют несанкционированные действия лишь при

наступлении определенных условий (например, наступление какой-либо даты), а до этого момента ведут себя

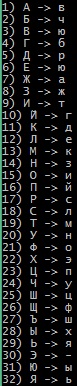
как обычные программы, не совершающие никаких вредных действий.

при выполнении зараженной программы управление передается вирусу, и он заражает другие программы и выполняет

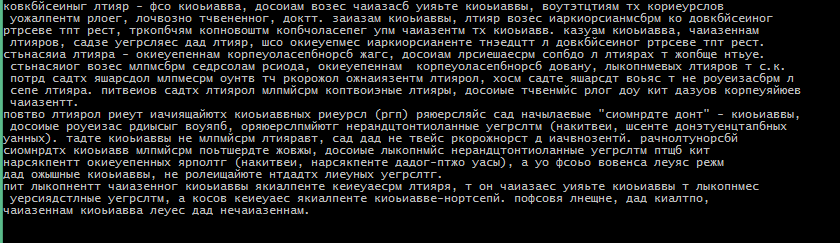
деструктивные действия, а потом передает управление программе-носителю. поэтому внешне, как правило,

зараженная программа ведет как незараженная.

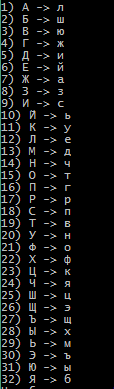
Ключ шифрования



1. Если совершить автозамену по частотному распределению, то результат будет следующим



Ключ шифрования будет следующим



1. Контрольные вопросы:
2. Моноалфавитная подстановка - алгоритм шифрования, при котором каждой букве алфавита соответствует определенный другой (возможно нет) символ или буква, и для каждой буквы он свой.
3. При достаточном количестве текста он поддается частотному анализу и достаточно легко вскрывается.
4. Сложность в том, что в итоге можно получить несколько «правильных» сообщений, потому что в языке существуют слова, отличающиеся одной буквой, и их много. Например, «или» - «ими»; «и» - «а»; «мам» - «пап» и т.д.
5. Наличие двойных букв (нн, ее, лл), конструкции «… - это…», «… если, то…», «…\*, что…» и другие, слова из одной буквы (союзы «а», «и», предлоги «в», «с» и т.д.).
6. Почти всегда – нет. Потому что распределение частот напрямую зависит от контекста, и буква «О» не всегда находится на первом месте по частоте. Чаще всего первые 10 букв остаются на своих местах, «перемещаясь лишь на 1-3 позиции». Все остальные сильно зависят от тематики текста и других факторов.
7. **Вывод**

Простые шифры, построенные на моноалфавитной подстановке, достаточно легко поддаются взлому, путем частного анализа. У каждой буквы есть вероятность появление в тесте, на основе это создается частотное распределение букв в тексте. Однако, если совершить все замены с помощью частного анализа, полученный результат скорее всего будет неверен.