Московский государственный технический университет им. Н.Э. Баумана

Факультет
Кафедра

Курс

Отчет по лабораторной работе №7
«Обработка символьных строк»
Вариант №4

Выполнил:
студент группы:

Подпись и дата:

Проверил:
преподаватель каф. ИУ5

Подпись и дата:

Москва, г.

Постановка задачи:

Провести кодирование и декодирование текста (массива символов) при помощи кода Цезаря с переменным сдвигом по таблице ASCII-кодов. Величина сдвига для каждой позиции в исходном тексте - сумма (по модулю 128) кодов символов слова кодового блокнота, стоящего в блокноте на той же позиции. Если кодовый блокнот имеет слов меньше, чем количество символов в исходном тексте, то по исчерпании слов в нём перейти к первому слову и продолжить.

Исследовать повторяемость символов в закодированном тексте (сколько каких кодов одного и того же исходного символа получено) в зависимости от кодового блокнота и длины исходного текста. Результаты исследования представить в виде таблицы (продумать формат таблицы самостоятельно). В таблице отобразить 5 неповторяющихся символов, выбранных случайным образом. Статистические данные хранить в массиве int stat[128].

Разработка алгоритма:

На основе кодового блокнота целесообразно сначала сформировать по заданному правилу целочисленный массив ключей, который затем использовать при кодировании. Эти действия можно оформить в виде отдельной функции keys(). Для кодирования текста можно использовать формулу: x= (y + k)%n
Где:
x – закодированный символ;
y – кодируемый символ;
k – ключ;
n – алфавит(в нашем случае 128);
Так же в случае если получаемый код по ASCII отрицательный, его нужно перевести в положительный.

FILE *dir, *dir1, *dir3, *dir4 – файлы key.txt, source.txt, encoded.txt, decoded.txt
char mass1_codir[1024] – массив, хранящий закодированные символы
char mass2_decodir[300] – массив, хранящий де-кодированные символы
int mass_keys[300] – массив ключей
int i - счетчик
int mass_dlya_povtora[5] – массив для вывода повторяющихся значений
char * tk – разбиение исходного текста по пробелам
char r – промежуточная переменная, копия tk[i]
int sum – сумма букв в слове
int j - счетчик
char m[200] - массив для подсчета повторяющихся значений
int count – кол-во символов

Текст программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include "caesar.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <locale.h>

void keys(char* t, int* mass_keys, int* n)// функция для подсчета суммы символов для каждого слова кодового блокнота
{
   char* tk, r;
```

```
    int sum;
    tk = strtok(t, " ");
    while (tk != NULL)
    {
        sum = 0;
        for (int i = 0; i < strlen(tk); i++)
        {
            r = (int)tk[i];
            sum += r;
        }
        mass_keys[n[0]] = sum;
        tk = strtok(NULL, " ");
        n[0] += 1;
    }
}
void codirovanie(char* t, char* mass1_codir, int* mass_keys, int* n)// функ. Для кодирования
{
    char* tk;
    int r;
    tk = strtok(t, " ");
    while (tk != NULL)
    {

        for (int i = 0; i < strlen(tk); i++)
        {
            if (n[2] >= n[0])
            {
                n[2] = 0;
            }
            if (tk[i] == '\n')
            {
                mass1_codir[n[1]] = '`';
            }
            else {
                r = tk[i] + mass_keys[n[2]];
                r = r % 128;
                if (r < 0) { r += 128; }
                mass1_codir[n[1]] = r;
                n[2] += 1;
            }
            n[1] += 1;
        }
        if (mass1_codir[n[1] - 1] != '`')
        {
            mass1_codir[n[1]] = '#';
            n[1] += 1;
        }
        tk = strtok(NULL, " ");
    }
}
void decodirovanie(char* mass1_codir, int* mass_keys, char* mass2_decodir, int* n)// функ.
декодирования
{
    char r;
    n[2] = 0;
```

```
    for (int i = 0; i < n[1]; i++)
    {
        if (mass1_codir[i] == '#')
        {
            mass2_decodir[i] = ' ';
        }
        else if (mass1_codir[i] == '`')
        {
            mass2_decodir[i] = '\n';
        }
        else {
            if (n[2] >= n[0])
            {
                n[2] = 0;
            }
            r = (int)(mass1_codir[i] - mass_keys[n[2]]) % 128;
            if (r < 0) { r += 128; }
            mass2_decodir[i] = r;
            n[2] += 1;
        }


    }
}
int Povtor1(int* dlya_povtora1, char* mass1_codir, int count)// функ. Подсчета повторяющихся элементов
{
    int i, j = 0, t = 0, c;
    char m[200];
    for (i = 0; i < count; i++)
    {
        c = mass1_codir[dlya_povtora1[i]];
        if (memchr(m, (char)mass1_codir[dlya_povtora1[i]], sizeof(m)) == NULL)
        {
            t += 1;
            m[j] = mass1_codir[dlya_povtora1[i]];
            j++;
        }
    }
    return t;
}
#define COLUMN_NUMBER 4 // число столбцов таблицы

struct I_print {        // данные для печати результатов интегрирования
    char symbol;        // название функции
    int kol_kod;        // значение интегральной суммы
    int symbol_v_tex;       // точное значение интеграла
    int slov_v_note;            // число разбиений области интегрирования  при котором достигнута требуемая точность
};

void printDividingLine(const char horizontalSybmol, const char connectorSybmol, const int m, const int *wn) {
    putchar(connectorSybmol);
```

```
    for (int line_i = 0; line_i < m; line_i++) {
        for (int line_j = 0; line_j < wn[line_i]; line_j++) {
            putchar(horizontalSybmol);
        }
        putchar(connectorSybmol);
    }
    putchar('\n');
}

void PrintTabl(struct I_print i_prn[], int k) {
    const char SIDE_SYBMOL = '|';
    const char HORIZONTAL_SYBMOL = '-';
    const char CONNECTOR_SYBMOL = '+';

    int wn[COLUMN_NUMBER] = {12, 18, 17, 15};   // ширина столбцов таблицы
    char *title[COLUMN_NUMBER] = {(char *) "Символ", (char *) "кол-во кодов", (char *) "символов в
тексте",
                                  (char *) "слов в блокноте"};
    int size[COLUMN_NUMBER];
    for (int i = 0; i < COLUMN_NUMBER; i++) {
        size[i] = (int) strlen(title[i]);
    }

    // шапка таблицы

    putchar(SIDE_SYBMOL);
    for (int line_i = 0; line_i < COLUMN_NUMBER; line_i++) {
        int half = (wn[line_i] - size[line_i]) / 2;
        for (int line_j = 0; line_j < half; line_j++) {
            putchar(' ');
        }
        printf("%s", title[line_i]);
        for (int line_j = 0; line_j < half; line_j++) {
            putchar(' ');
        }
        putchar(SIDE_SYBMOL);
    }
    putchar('\n');

    printDividingLine(HORIZONTAL_SYBMOL, CONNECTOR_SYBMOL, COLUMN_NUMBER, wn);

    // заполнение таблицы
    for (int i = 0; i < k; i++) {
        putchar(SIDE_SYBMOL);
        printf("%12c", i_prn[i].symbol);
        putchar(SIDE_SYBMOL);

        char kol_kod[15];
        printf("%18d", i_prn[i].kol_kod);
        putchar(SIDE_SYBMOL);

        char symbol_v_tex[15];
        printf("%17d", i_prn[i].symbol_v_tex);
        putchar(SIDE_SYBMOL);
```

```
        printf("%15d", i_prn[i].slov_v_note);
        putchar(SIDE_SYBMOL);
        putchar('\n');
    }
}


int main() {
    setlocale(LC_ALL, "Russian");
    FILE *dir, *dir1, *dir3, *dir4;
    char t[200], mass1_codir[1024], mass2_decodir[300];
    int mass_keys[300], i, mass_dlya_povtora[5] = {0, 0, 0, 0, 0}, l[5], n[3] = {0, 0, 0};
    dir = fopen("key.txt", "r");
    dir1 = fopen("source.txt", "r");
    dir3= fopen("encoded.txt", "w");
    dir4= fopen("decoded.txt", "w");
    printf("содержимое key.txt:\n");
    while (!feof(dir))
    {
        fgets(t, 199, dir);
        printf("%s", t);
        keys(t, mass_keys, n);

    }
    printf("\n\nсодержимое source.txt:\n");
    while (!feof(dir1))
    {
        fgets(t, 199, dir1);
        printf("%s", t);
        codirovanie(t, mass1_codir, mass_keys, n);
    }
    printf("\n\nсодержимое encoded.txt:\n");
    for (i = 3; i < n[1]; i++)
    {
        if (mass1_codir[i] == '#')
        {
            fprintf(dir3, "%c", ' ');
            printf("%c", ' ');
        }
        else if (mass1_codir[i] == '`')
        {
            fprintf(dir3, "%c", '\n');
            printf("%c", '\n');
        }
        else
        {
            printf("%c", mass1_codir[i]);
            fprintf(dir3, "%c", mass1_codir[i]);
        }
    }
    decodirovanie(mass1_codir, mass_keys, mass2_decodir, n);
    printf("\n\nсодержимое decoded.txt:\n");
    for (i = 3; i < n[1]; i++)
    {
        fprintf(dir4, "%c", mass2_decodir[i]);
```

```
        printf("%c", mass2_decodir[i]);
    }
    printf("\n\n");
    i = 0;
    // нахождение 5 случайных символов
    char r;
    while (i < 5)
    {
        r = (char)mass2_decodir[3+rand() % (n[1] - 3+1)];
        if (r != ' ' && memchr(mass_dlya_povtora, r, sizeof(mass_dlya_povtora)) == NULL && r != '\n' )
        {
            mass_dlya_povtora[i] = (int)r;
            i++;
        }
    }
    // кол-во кодов
    int count, dlya_povtora1[200], mass_count[5];
    for (int j = 0; j < 5; j++) {
        count = 0;
        for (i = 0; i < n[1]; i++) {
            if (mass_dlya_povtora[j] == mass2_decodir[i]) {
                dlya_povtora1[count] = i;
                count++;
            }
        }
        mass_count[j] = count;
        //printf(" %c", mass_dlya_povtora[j]);
        l[j] = Povtor1(dlya_povtora1, mass1_codir, count);
        //printf(" %d", l[j]);
    }
    fclose(dir);
    fclose(dir1);
    fclose(dir3);
    fclose(dir4);
    //значения для таблицы
    struct I_print struc;
    struct I_print f_Rect[5];
    for (i = 0; i < 5; i++) {
        struc.symbol = mass_dlya_povtora[i];
        struc.kol_kod = l[i];
        struc.symbol_v_tex = mass_count[i];
        struc.slov_v_note = n[0];
        f_Rect[i] = struc;

    }
    PrintTabl(f_Rect, 5);
}
```

Анализ результатов:

```
содержимое key.txt:
The only memory left is trauma
Imaginary friend's kind words
The evening train was shaking
I purified the imperfect flowers
The pain in my heart getting higher
My comedy show at its peak
The frogs were crying on our way home
This is my last war

содержимое source.txt:
Let's start a new life from the darkness
Until the light reveals the end
Sinister faces, growing curses
This is my last war
This

содержимое encoded.txt:
RO §s↑- ☻ Z♥B [2>& %~►§ KNy Vb8l/:C▲
v☼'uI J9↑ ♦EM└> ‼'☼►=¶ ♠♫▲ ♠Z☻
▲X7A43q‼ ♫8Iye- -s0L9 ♦(~P;D
EY ↔= ♫; ♣♀ ↑s↑
       U◄

содержимое decoded.txt:
's start a new life from the darkness
Until the light reveals the end
Sinister faces, growing curses
This is my last war
This

|   Символ   |   кол-во кодов   |символов в тексте|слов в блокноте|
+------------+------------------+-----------------+---------------+
|           i|                 8|                9|             46|
|           t|                10|               10|             46|
|           ,|                 1|                1|             46|
|           r|                 5|                8|             46|
|           l|                 5|                5|             46|
```

Рис.1. Пример работы программы

Источники:

1. Язык программирования C [2015] Брайан У. Керниган, Деннис М. Ритчи