ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИСCЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

**Дисциплина:**

«Программирование»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**Вариант 2-28**

**Выполнил:**

Студент гр. N3151, Мочеков Семён Сергеевич



**Проверил:**

Гирик Алексей Валерьевич

Санкт-Петербург

2023

1. **Задание.**

В одномерном массиве uint32\_t, содержащем столбцы матрицы, провести преобразования:

Назовем характеристикой строки результат поразрядной операции "штрих Шеффера" между элементами строки. Осуществить циклический сдвиг каждой строки на количество единиц в пяти старших битах значения характеристики.

1. **Makefile**

.PHONY: all clean

APP=lab2ssmN3151

CFLAGS=-Wall -Wextra -Werror -g

all: $(APP)

$(APP):

gcc -o $(APP) $(CFLAGS) $(APP).c

clean:

rm $(APP)

1. **Работа алгоритма**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание







**IV. Исходный код алгоритма**

#include <stdio.h>

#include <inttypes.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#define pint uint32\_t

#define spec "%" SCNu32

// назовём возвращаемые коды

typedef enum {

    no\_rand, yee\_rand

} is\_matrix\_rand ;

typedef enum {

    OK, Word, Overflow

} num\_errs ;

// функция проверки числа

num\_errs num\_check(char\* arg){

    size\_t len = strlen(arg);

    char\* end;

    long int value = strtol(arg, &end, 10);

    if (end != arg + len) {

        return Word;

    }

    if (len > 10) {

        return Overflow;

    }

    if (value < 0 || value > UINT32\_MAX) {

        return Overflow;

    }

    return OK;

}

// функция вычисления характеристики строки

pint shiffer\_rank(pint \*arr, pint str, pint sdv, pint diapoz){

    pint k\_edin = 0, shiff = arr[str], e = 1 << (sizeof(arr[str]) \* 8 - 1);

    for (pint i = 1; i < sdv; ++i){

        shiff = ~(shiff & arr[i \* sdv + str]);

    }

    for (pint i = 0; i < diapoz; ++i){

        k\_edin += (shiff & (e >>(1 \* i))) != 0;

    }

    return k\_edin;

}

// функция циклического сдвига строк

void sdvig\_memory\_eater(pint \*arr, pint str, pint stb, pint diapoz){

    pint \*pozit = malloc(stb \* str \* sizeof(\*pozit)), exp;

    for (pint j = 0; j < str; ++j){

        exp = shiffer\_rank(arr, j, str, diapoz);

        for (pint i = 0; i < stb; ++i){

            pozit[((i+exp) % stb) \* str + j]= arr[i \* str + j];

        }

    }

    for (pint i = 0; i < stb\*str; ++i){

        arr[i] = pozit[i];

    }

    free(pozit);

}

int main(int argc, char \*argv[]){

    srand(time(NULL));

    const char \*env = getenv("DEBUG");

    pint str = 1, stb = 1;

    is\_matrix\_rand param;

    // анализ входных данных на запуске

    if (argc > 4){ printf("Ошибка: слишком много аргументов\n"); return 0; }

    switch(argc) {

        case 1:

            printf("Ошибка: отсутствуют обязательные аргументы\n"); return 0;

        case 2:

            printf("Ошибка: неполный ввод агрументов\n"); return 0;

        case 3:

            num\_errs t1 = num\_check(argv[1]), t2 = num\_check(argv[2]);

            if (argv[1][0] == '-'){ printf("Ошибка: неполный ввод\n"); return 0; }

            if (t1 == OK){ sscanf(argv[1], spec, &str); }

            else if (t1 == Overflow) { printf("Ошибка: \'%s..\' выходит за допустимые значения\n", argv[1]); return 0; }

            else { printf("Ошибка: \'%s\' не является числом\n", argv[1]); return 0; }

            if (t2 == OK) { sscanf(argv[2], spec, &stb); }

            else if (t2 == Overflow) { printf("Ошибка: \'%s..\' выходит за допустимые значения\n", argv[2]); return 0; }

            else { printf("Ошибка: \'%s..\' не является числом\n", argv[2]); return 0; }

            param = yee\_rand;

            break;

        case 4:

            num\_errs t3 = num\_check(argv[2]), t4 = num\_check(argv[3]);

            if (!((argv[1][0] == '-') && (argv[1][1] == 'm') && (argv[1][2] == 0))){ printf("Ошибка: \'%s\' не поддерживается\n", argv[1]); return 0;}

            if (t3 == OK){ sscanf(argv[2], spec, &str); }

            else if (t3 == Overflow) { printf("Ошибка: \'%s..\' выходит за допустимые значения\n", argv[2]); return 0; }

            else { printf("Ошибка: \'%s\' не является числом\n", argv[2]); return 0; }

            if (t4 == OK){ sscanf(argv[3], spec, &stb); }

            else if (t4 == Overflow) { printf("Ошибка: \'%s..\' выходит за допустимые значения\n", argv[3]); return 0; }

            else { printf("Ошибка: \'%s..\' не является числом\n", argv[3]); return 0; }

            param = no\_rand;

            break;

    }

    // инициализация матрицы

    pint \*array = malloc(sizeof(\*array) \* str \* stb);

    if (array == NULL) { printf("Memory error\n"); return -1; }

    if (param == no\_rand) {

        num\_errs flag = OK, nc;

        char\* ss = malloc(128);

        char\* errs = malloc(128);

        printf("Введите строки матрицы:\n");

        for (pint j = 0; j < str; ++j){

            for (pint i = 0; i < stb; ++i){

                scanf("%s", ss);

                nc = num\_check(ss);

                if (!flag && !nc) { sscanf(ss, spec, &array[i \* str + j]); }

                else if (!flag){

                    memmove(errs, ss, 128);

                    if (nc == Word) { flag = Word; }

                    if (nc == Overflow) { flag = Overflow; }

                }

            }

            if (flag == Word){ printf("Ошибка: \'%s..\' не является числом\n", errs); return 0; }

            if (flag == Overflow){ printf("Ошибка: \'%s..\' выходит за допустимые значения\n", errs); return 0; }

        }

        free(ss);

        free(errs);

    } else if (param == yee\_rand) {

        for (pint i = 0; i < str\*stb; ++i) {

            array[i] = rand();

        }

    }

    // вывод данной матрицы

    printf("Исходная матрица:\n");

    if (env) {

        for (pint j = 0; j < str; ++j){

            for (pint i = 0; i < stb; ++i){

                printf(spec " ", array[i \* str + j]);

            }

            printf("Характеристика = " spec "\n", shiffer\_rank(array, j, str, 5));

        }

    }else {

        for (pint j = 0; j < str; ++j){

            for (pint i = 0; i < stb; ++i){

                printf(spec " ", array[i \* str + j]);

            }

            printf("\n");

        }

    }

    // сдвиг

    sdvig\_memory\_eater(array, str, stb, 5);

    // вывод результат

    printf("Результат:\n");

    for (pint j = 0; j < str; ++j){

        for (pint i = 0; i < stb; ++i){

            printf(spec " ", array[i \* str + j]);

        }

        printf("\n");

    }

    // no-spy

    explicit\_bzero(array, str \* stb \* sizeof(\*array));

    free(array);

    return 0;

}