МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)**»**
Центр образовательных программ топ-уровня в сфере информационных технологий Образовательного центра Института №8

КУРСОВАЯ РАБОТА

по дисциплине «Фундаментальная информатика»

на тему: **«Теоретические основы алгоритмизации и практика программирования на языке С»**

Выполнил:

студент гр. М8О-101БВ-25

Козлов Сергей Сергеевич

Проверил: Крылов С.С.

Оценка:

Дата:

Москва 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная работа подводит итог лабораторному практикуму в рамках курса «Фундаментальная информатика». Её основная задача — трансформация теоретических знаний в практические умения: от алгоритмизации до написания кода на языке C. При решении задач особое внимание уделялось балансу между пониманием абстрактных моделей вычислений и прикладными приёмами работы с данными на низком уровне.

Каждая лабораторная работа была посвящена отдельной проблеме, благодаря чему освоение материала происходило поэтапно.

В ходе практики были затронуты следующие темы: базовое взаимодействие с ОС Linux, фундаментальные основы вычислений (включая машины Тьюринга и алгоритмы Маркова), программирование с использованием битовых операций, арифметика целых чисел, обработка матриц и реализация конечных автоматов (FSM).

1 ПРАКТИЧЕСКАЯ ЧАСТЬ

1.1 Лабораторная работа №1

Тема: Утилиты командной строки Linux.

Условие: выполнить набор операций по администрированию файловой системы в Linux: массовое переименование файлов, создание серии файлов по шаблону, сжатие архивов, перемещение файлов относительными путями, поиск по размеру и типу, фильтрация текста и автоматизация удаления пустых файлов через Bash-скрипт.

Ход работы: Деятельность представляет собой последовательный запуск команд интерпретатора Linux для манипуляции файлами и папками в рабочей директории. Каждое действие предназначено для решения уникальной задачи: переименование объектов, формирование древовидной структуры, регулировка прав доступа для групп лиц, фильтрационное копирование и удаление, а также поиск объектов по маске, расширению, атрибутам и глубине. Исполнение поручений нуждается в внимательном применении опций утилит mv, mkdir, chmod, rm, cp и find, дабы не затронуть сторонние файлы и каталоги.

Код программы приведен в Приложении А.

1.2 Лабораторная работа №2

Тема: Диаграммы Тьюринга.

Условие: разработать машину Тьюринга для перевода числа из шестнадцатеричной системы счисления в четверичную, обеспечив логарифмическую сложность вычислений.

Ход работы: была спроектирована диаграмма машины Тьюринга, реализующая перевод чисел между указанными системами счисления. Алгоритм основан на свойстве вложенности оснований: поскольку основание исходной системы является степенью основания целевой системы, каждая цифра входного числа заменяется фиксированной последовательностью цифр результата. Машина последовательно обрабатывает входную строку,

считывая символы и подставляя вместо них соответствующие группы символов новой системы. Такой подход позволяет выполнить преобразование за время, пропорциональное количеству цифр в записи числа, что соответствует требуемой логарифмической сложности.

Диаграмма машины Тьюринга приведена в Приложении Б.

1.3 Лабораторная работа №3

Тема: Конструирование нормальных алгоритмов Маркова.

Условие: разработать нормальный алгоритм Маркова для вычисления суммы двух неотрицательных целых чисел, записанных в троичной системе счисления и разделенных знаком сложения.

Ход работы: был составлен нормальный алгоритм Маркова, реализующий поразрядное сложение двух троичных чисел. Алгоритм обрабатывает входную строку справа налево, последовательно суммируя соответствующие разряды с учетом возможного переноса в старший разряд. Правила подстановки описывают все возможные комбинации складываемых цифр и переноса, заменяя их на результат суммы и новый флаг переноса. По завершении обработки всех разрядов остаточный перенос дописывается в начало результата, а служебные символы удаляются.

Правила нормального алгоритма Маркова приведены в Приложении В.

1.4 Лабораторная работа №4

Тема: Целые числа и системы счисления.

Условие: реализовать алгоритм, который после каждых двух цифр десятичной записи числа вставляет их сумму, если результат является однозначным числом.

Ход работы: был реализован алгоритм посимвольной обработки строки, представляющей число. Программа последовательно обрабатывает цифры входной строки, группируя их в непересекающиеся пары. Для каждой такой пары вычисляется сумма, и, если она является однозначным числом (не превышает 9), результат вставляется сразу после второй цифры пары.

Код программы приведен в Приложении Г.

1.5 Лабораторная работа №5

Тема: Операции над битовыми множествами.

Условие: определить, содержится ли во входном тексте хотя бы одно слово, все согласные буквы которого являются глухими.

Ход работы: был реализован алгоритм посимвольного анализа входного текста. Для эффективной проверки использовалось битовое множество, хранящее маску глухих согласных букв. При обработке слова каждая согласная буква проверялась на принадлежность к этому множеству. Если встречалась звонкая согласная, слово исключалось из рассмотрения. Если все согласные в слове оказывались глухими, условие считалось выполненным. Алгоритм корректно обрабатывал границы слов и игнорировал регистр букв. При нахождении подходящего слова выводился утвердительный ответ, иначе отрицательный.

Код программы приведен в Приложении Д.

1.6 Лабораторная работа №6

Тема: Обработка матричных данных и циклические сдвиги.

Условие: выполнить циклический сдвиг элементов квадратной матрицы вдоль спиральной траектории против часовой стрелки на количество позиций, равное номеру группы.

Ход работы: в ходе работы была написана программа на языке C. Для хранения данных использовалась динамическая память, размер которой зависел от входных данных. Реализован обход элементов матрицы по спирали против часовой стрелки, при котором значения сохраняются во временный одномерный массив. Затем выполняется сдвиг элементов массива на заданное количество позиций с операцией остатка для зацикливания. Полученные значения записываются обратно в матрицу. В конце работы вся выделенная память освобождается.

Код программы приведен в Приложении Е.

1.7 Лабораторная работа №7

Тема: Конечные автоматы и регулярные выражения.

Условие: выделить все шестнадцатеричные числа во входном тексте, длина которых соответствует максимальному размеру целого числа в тридцатидвухразрядной архитектуре.

Ход работы: в ходе работы была написана программа на языке C. Для распознавания чисел использовался конечный автомат, обрабатывающий входной поток посимвольно. Автомат выявляет последовательности с префиксом шестнадцатеричной системы и подсчитывает количество цифр. Если длина числа достигает предельного значения для заданной разрядности, оно выводится в результат. Числа с другим количеством цифр игнорируются. Предусмотрена корректная обработка границ слов и некорректных символов.

Код программы приведен в Приложении Ж.

ЗАКЛЮЧЕНИЕ

Выполнение цикла лабораторных работ позволило закрепить теоретические знания по дисциплине «Фундаментальная информатика» и применить их на практике. Тематика заданий охватывала широкий спектр вопросов: от базовых операций в операционной системе Linux до разработки программ на языке C с использованием низкоуровневых механизмов.

В ходе практики были освоены методы работы с командной оболочкой, созданы модели вычислительных процессов (машины Тьюринга, алгоритмы Маркова) и реализованы алгоритмы обработки различных структур данных. Особый акцент делался на эффективном использовании памяти, битовых операциях и построении конечных автоматов для анализа текста.

Можно констатировать, что цели курса достигнуты. Полученные компетенции в области алгоритмизации и системного программирования станут хорошей базой для дальнейшего обучения и решения профессиональных задач в сфере информационных технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Metanit.com. Руководство по C [Электронный ресурс]. — URL: https://metanit.com/c (дата обращения: 11.01.2025).

2. Хопкрофт Дж., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. — М.: Вильямс, 2019.

3. cppreference.com. Справочник по стандартной библиотеке C [Электронный ресурс]. — URL: https://en.cppreference.com/w/c(дата обращения: 18.01.2025).

ПРИЛОЖЕНИЕ А

Код лабораторной работы №1

Код решений представлены ниже.

```
# Изменить расширение всех txt файлов в формат md
for f in *.txt; do mv "$f" "${f%.txt}.md"; done

# Создать файлы формата txt, в которых название - одна каждая буква латинского алфавита:
# Example: a.txt, b.txt... z.txt
touch {a..z}.txt

# Сожмите архив trash.tar с максимальной степенью сжатия
gzip -9 trash.tar

#Переместите файл me.txt в папку home.
mv me.txt ../../chill/home

# Выведите топ 7 самых тяжёлых файлов в диапазоне от 1 до 100 MB
find . -type f -size +1M -size -100M -printf '%s %p\n' | sort -rn | head -7

# Есть файл с паролями в формате:
# название сайта : пароль
# Найти пароль от сайта vk.com.
grep 'vk\.com' passwords.txt | sed 's/.*: *//'

# Найди мне все файлы формата *mp4, которые не являются ремиксами.
# Файл считается ремиксом, если в названии встречается слово "remix".
find . -type f -name "*.mp4" ! -iname "*remix*"

# Найти номера строк, на которых находится слово def
grep -nw 'def' filename

# Найти все директории, начиная с текущей
find . -type d

# Написать bash скрипт, удаляющий пустые файлы в текущей директории
find . -maxdepth 1 -type f -empty -delete
```

Рисунок А — команды для решения задач.

ПРИЛОЖЕНИЕ Б

Код лабораторной работы №2

Ниже представлены рисунки диаграмм машин Тьюринга.

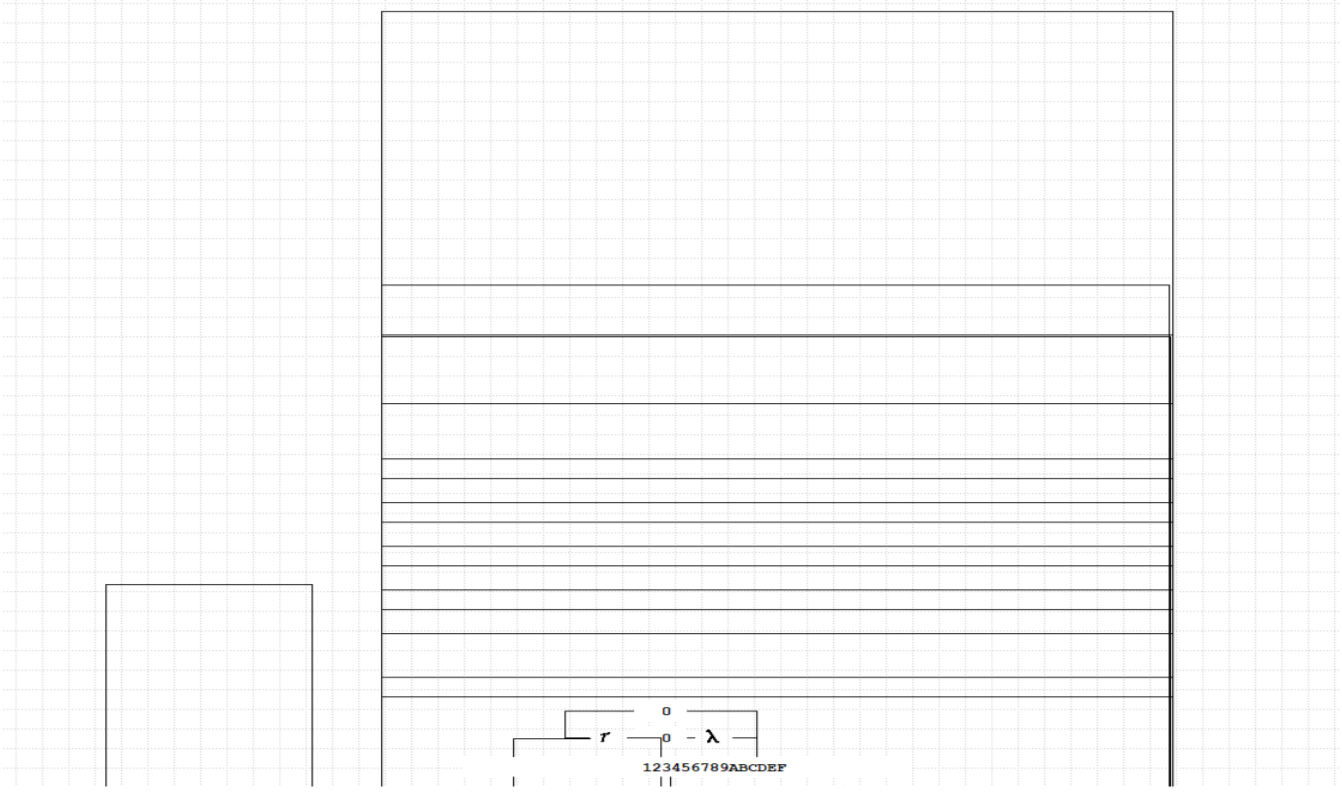


Рисунок Б.1 — Диаграмма машины Тьюринга.

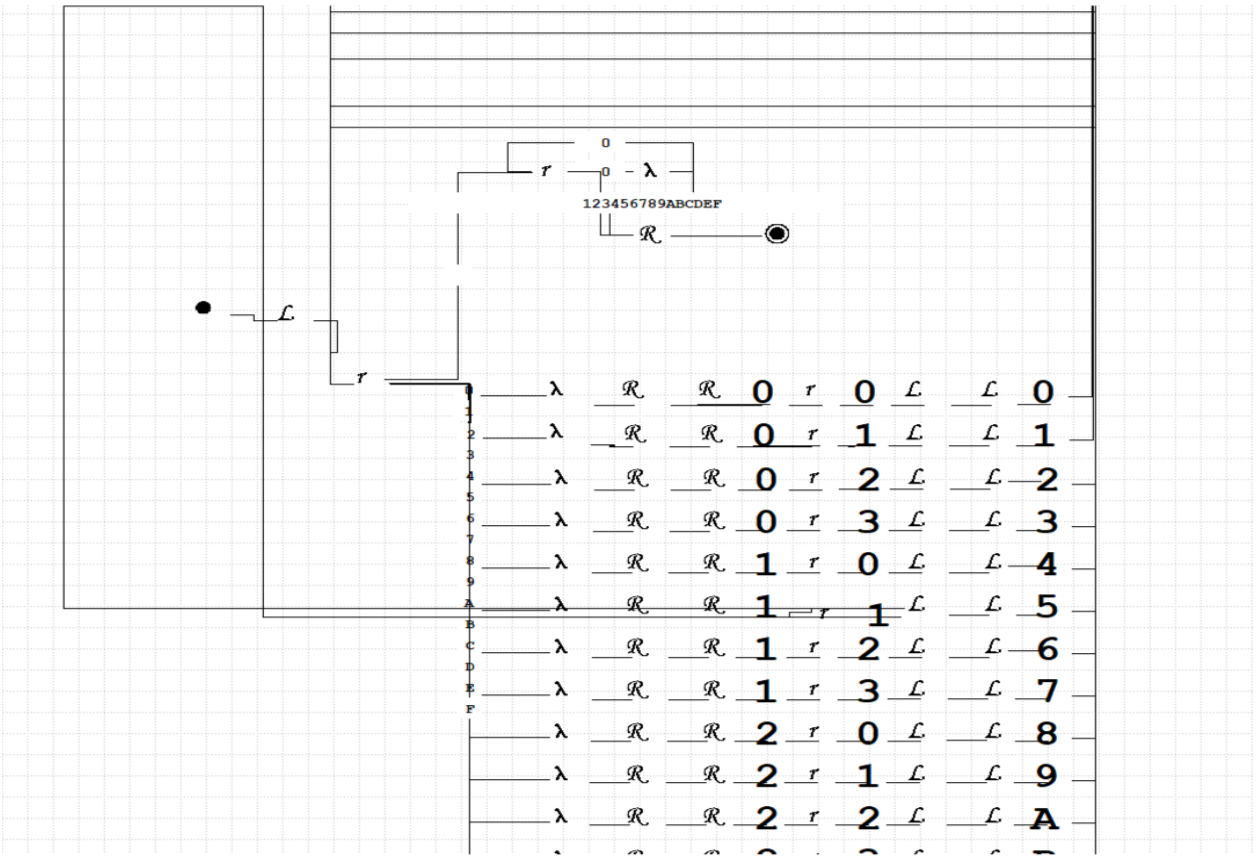


Рисунок Б.2 — Продолжение диаграммы машины Тьюринга.

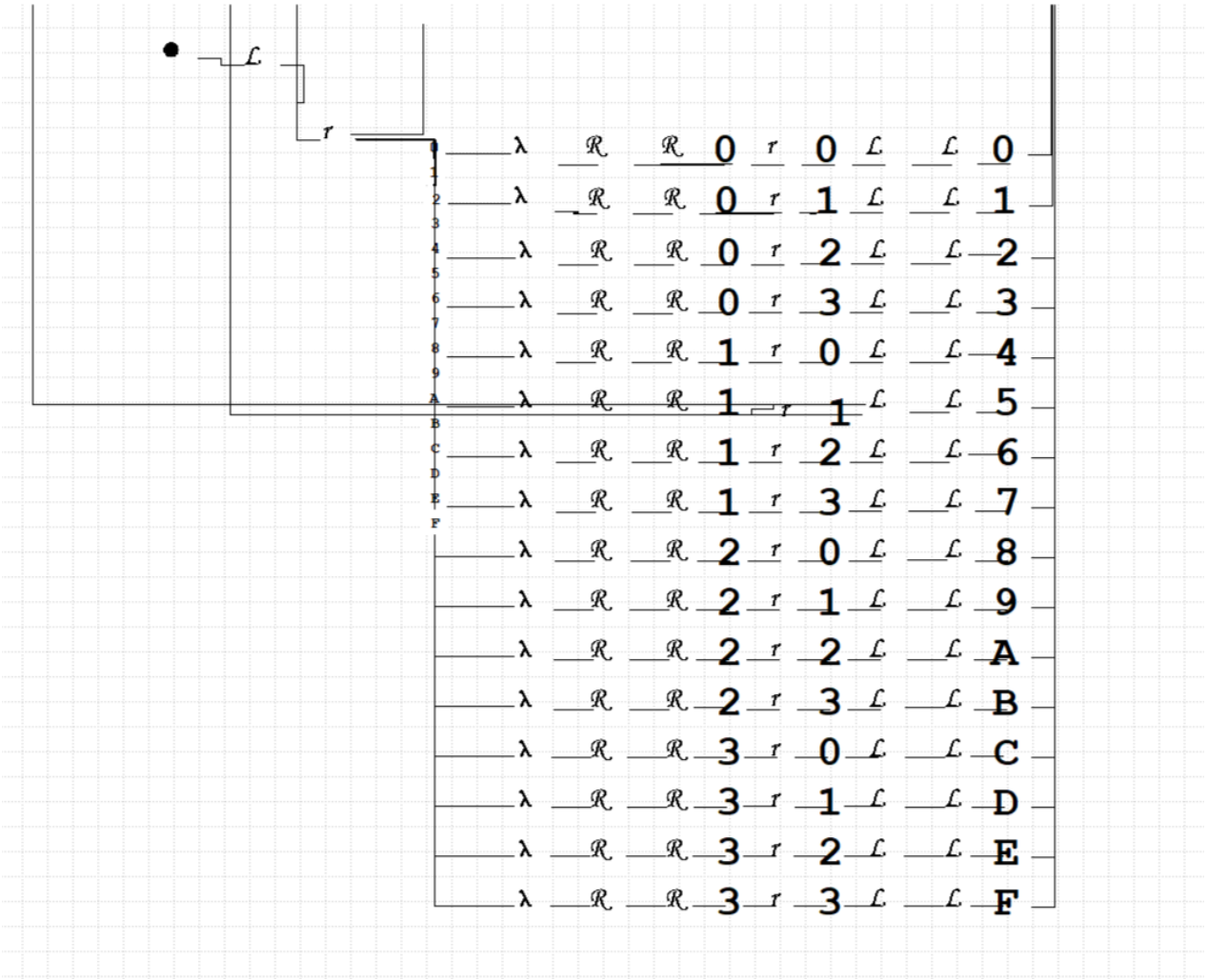


Рисунок Б.3 — Продолжение диаграммы машины Тьюринга.

ПРИЛОЖЕНИЕ В

Код лабораторной работы №3

Правила представлены ниже:

+ → +|

0|+0 → |0

0|+1 → |1

0|+2 → |2

1|+0 → |1

1|+1 → |2

2|+0 → |2

1|+2 → ^|0

2|+1 → ^|0

2|+2 → ^|1

0| → |0

1| → |1

2| → |2

^|0+0 → |1

^|0+1 → |2

^|0+2 → ^|0

^|1+0 → |2

^|1+1 → ^|0

^|1+2 → ^|1

^|2+0 → ^|0

^|2+1 → ^|1

^|2+2 → ^|2

^|+ → 1|

|+ → .

| → .

ПРИЛОЖЕНИЕ Г

Код лабораторной работы №4

Ниже представлен код решения лабораторной работы 4.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void) {
    char input[256];
    char output[512];

    printf("Введите число: ");
    if (fgets(input, sizeof(input), stdin) == NULL) {
        return 1;
    }

    size_t len = strlen(input);
    if (len > 0 && input[len - 1] == '\n') {
        input[len - 1] = '\0';
        len--;
    }

    size_t j = 0;
    int count = 0;

    for (size_t i = 0; i < len; i++) {
        if (input[i] >= '0' && input[i] <= '9') {
            output[j++] = input[i];
            count++;

            if (count == 2 && i + 1 < len) {
                int d1 = input[i - 1] - '0';
                int d2 = input[i] - '0';
                int sum = d1 + d2;

                if (sum >= 0 && sum <= 9) {
                    output[j++] = sum + '0';
                }
                count = 0;
            }
        } else {
            output[j++] = input[i];
            count = 0;
        }
    }

    output[j] = '\0';
    printf("Результат: %s\n", output);

    return 0;
}
```

Рисунок Г — Код решения.

ПРИЛОЖЕНИЕ Д

Код лабораторной работы №5

Ниже приведен код лабораторной работы номер 5.

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

#define MAX_WORD 256


unsigned int voiceless_mask = 0;


void init_voiceless_mask(void) {
    const char *voiceless = "кпстфхцчшщКПСТФХЦЧШЩ";
    for (int i = 0; voiceless[i]; i++) {
        voiceless_mask |= (1u << (voiceless[i] & 0x1F));
    }
}


bool is_voiceless_consonant(char c) {
    return (voiceless_mask & (1u << (c & 0x1F))) != 0;
}


bool is_consonant(char c) {
    const char *consonants = "бвгджзклмнпрйБВГДЖЗКЛМНПРЙ";
    for (int i = 0; consonants[i]; i++) {
        if (c == consonants[i]) return true;
    }
    return is_voiceless_consonant(c);
}


bool check_word(const char *word) {
    bool has_consonant = false;

    for (int i = 0; word[i]; i++) {
        if (is_consonant(word[i])) {
            has_consonant = true;
            if (!is_voiceless_consonant(word[i])) {
                return false;
            }
        }
    }

    return has_consonant;
}

int main(void) {
    init_voiceless_mask();

    char word[MAX_WORD];
    bool found = false;

    printf("Введите текст (завершите Ctrl+D или Ctrl+Z):\n");
```

Рисунок Д.1 — Код решения.

```c
bool is_consonant(char c) {
    }
    return is_voiceless_consonant(c);
}


bool check_word(const char *word) {
    bool has_consonant = false;

    for (int i = 0; word[i]; i++) {
        if (is_consonant(word[i])) {
            has_consonant = true;
            if (!is_voiceless_consonant(word[i])) {
                return false;
            }
        }
    }

    return has_consonant;
}

int main(void) {
    init_voiceless_mask();

    char word[MAX_WORD];
    bool found = false;

    printf("Введите текст (завершите Ctrl+D или Ctrl+Z):\n");

    while (scanf("%255s", word) == 1) {

        size_t len = strlen(word);
        size_t start = 0, end = len;

        while (start < end && !isalpha((unsigned char)word[start])) start++;
        while (end > start && !isalpha((unsigned char)word[end - 1])) end--;

        if (start < end) {
            word[end] = '\0';
            if (check_word(word + start)) {
                found = true;
                printf("Найдено слово: %s\n", word + start);
            }
        }
    }

    if (found) {
        printf("\nДА: есть слово, все согласные которого глухие.\n");
    } else {
        printf("\nНЕТ: таких слов не найдено.\n");
    }

    return 0;
}
```

Рисунок Д.2 — Код решения.

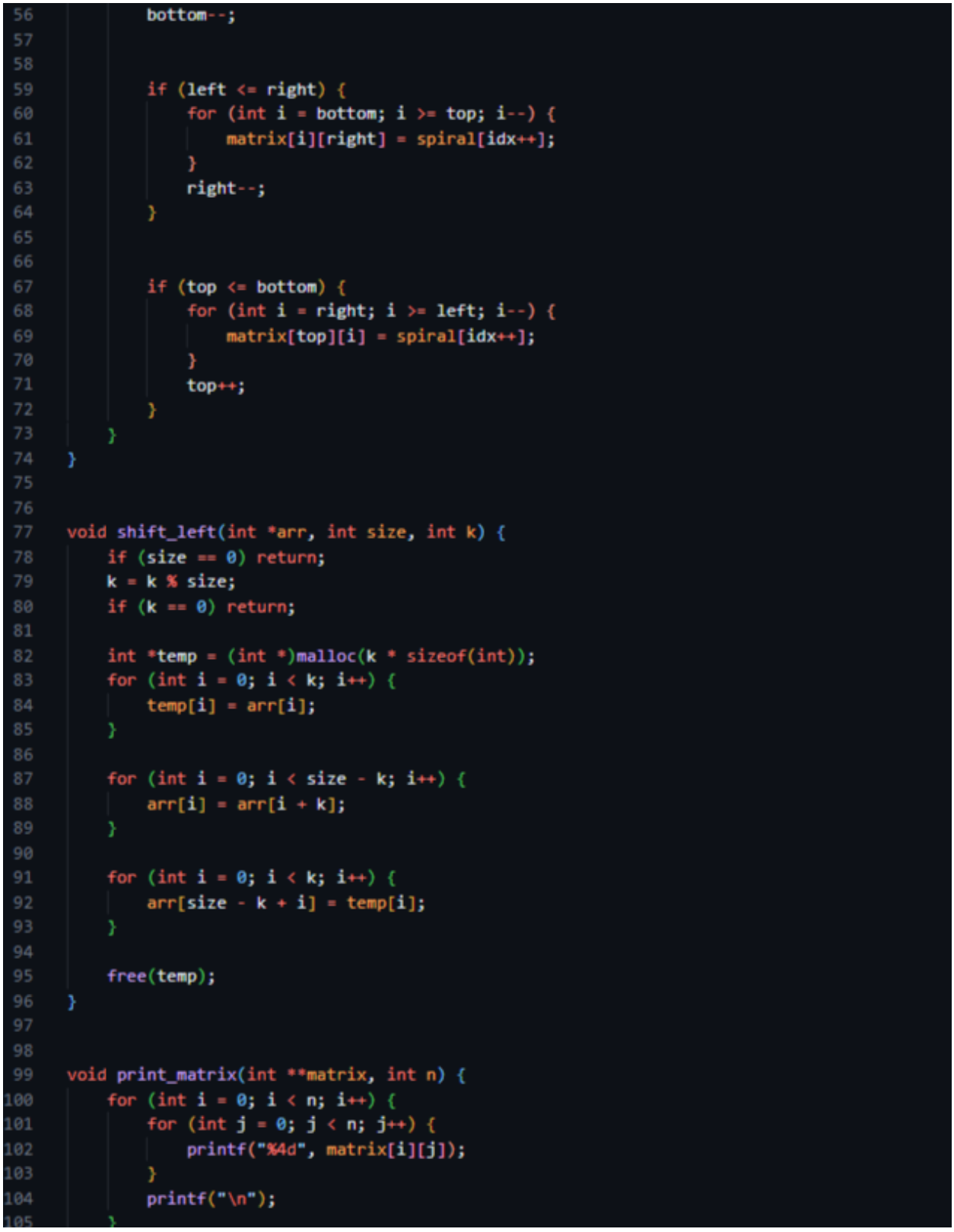
ПРИЛОЖЕНИЕ Е

Код лабораторной работы №6

Ниже приведен код лабораторной работы номер 6.

```c
#include <stdio.h>
#include <stdlib.h>


void get_spiral(int **matrix, int *spiral, int n) {
    int top = 0, bottom = n - 1, left = 0, right = n - 1;
    int idx = 0;

    while (top <= bottom && left <= right) {

        for (int i = top; i <= bottom; i++) {
            spiral[idx++] = matrix[i][left];
        }
        left++;


        for (int i = left; i <= right; i++) {
            spiral[idx++] = matrix[bottom][i];
        }
        bottom--;


        if (left <= right) {
            for (int i = bottom; i >= top; i--) {
                spiral[idx++] = matrix[i][right];
            }
            right--;
        }


        if (top <= bottom) {
            for (int i = right; i >= left; i--) {
                spiral[idx++] = matrix[top][i];
            }
            top++;
        }
    }
}


void set_spiral(int **matrix, int *spiral, int n) {
    int top = 0, bottom = n - 1, left = 0, right = n - 1;
    int idx = 0;

    while (top <= bottom && left <= right) {

        for (int i = top; i <= bottom; i++) {
            matrix[i][left] = spiral[idx++];
        }
        left++;


        for (int i = left; i <= right; i++) {
            matrix[bottom][i] = spiral[idx++];
        }
```

Рисунок Е.1 — Начало кода 6 лабораторной работы.



```
            bottom--;


            if (left <= right) {
                for (int i = bottom; i >= top; i--) {
                    matrix[i][right] = spiral[idx++];
                }
                right--;
            }


            if (top <= bottom) {
                for (int i = right; i >= left; i--) {
                    matrix[top][i] = spiral[idx++];
                }
                top++;
            }
        }
    }


    void shift_left(int *arr, int size, int k) {
        if (size == 0) return;
        k = k % size;
        if (k == 0) return;

        int *temp = (int *)malloc(k * sizeof(int));
        for (int i = 0; i < k; i++) {
            temp[i] = arr[i];
        }

        for (int i = 0; i < size - k; i++) {
            arr[i] = arr[i + k];
        }

        for (int i = 0; i < k; i++) {
            arr[size - k + i] = temp[i];
        }

        free(temp);
    }


    void print_matrix(int **matrix, int n) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                printf("%4d", matrix[i][j]);
            }
            printf("\n");
        }
```

Рисунок Е.2 — Продолжение кода 6 лабораторной работы.

```c
}

int main(void) {
    int n, group;

    printf("Введите размер матрицы: ");
    scanf("%d", &n);

    printf("Введите номер группы (количество сдвигов): ");
    scanf("%d", &group);


    int **matrix = (int **)malloc(n * sizeof(int *));
    for (int i = 0; i < n; i++) {
        matrix[i] = (int *)malloc(n * sizeof(int));
    }


    printf("Введите элементы матрицы (%d чисел):\n", n * n);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }


    int *spiral = (int *)malloc(n * n * sizeof(int));


    get_spiral(matrix, spiral, n);


    shift_left(spiral, n * n, group);


    set_spiral(matrix, spiral, n);


    printf("\nРезультат после сдвига на %d позиций:\n", group);
    print_matrix(matrix, n);


    free(spiral);
    for (int i = 0; i < n; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

Рисунок Е.3 — Продолжение кода 6 лабораторной работы.

ПРИЛОЖЕНИЕ Ж

Код лабораторной работы №7

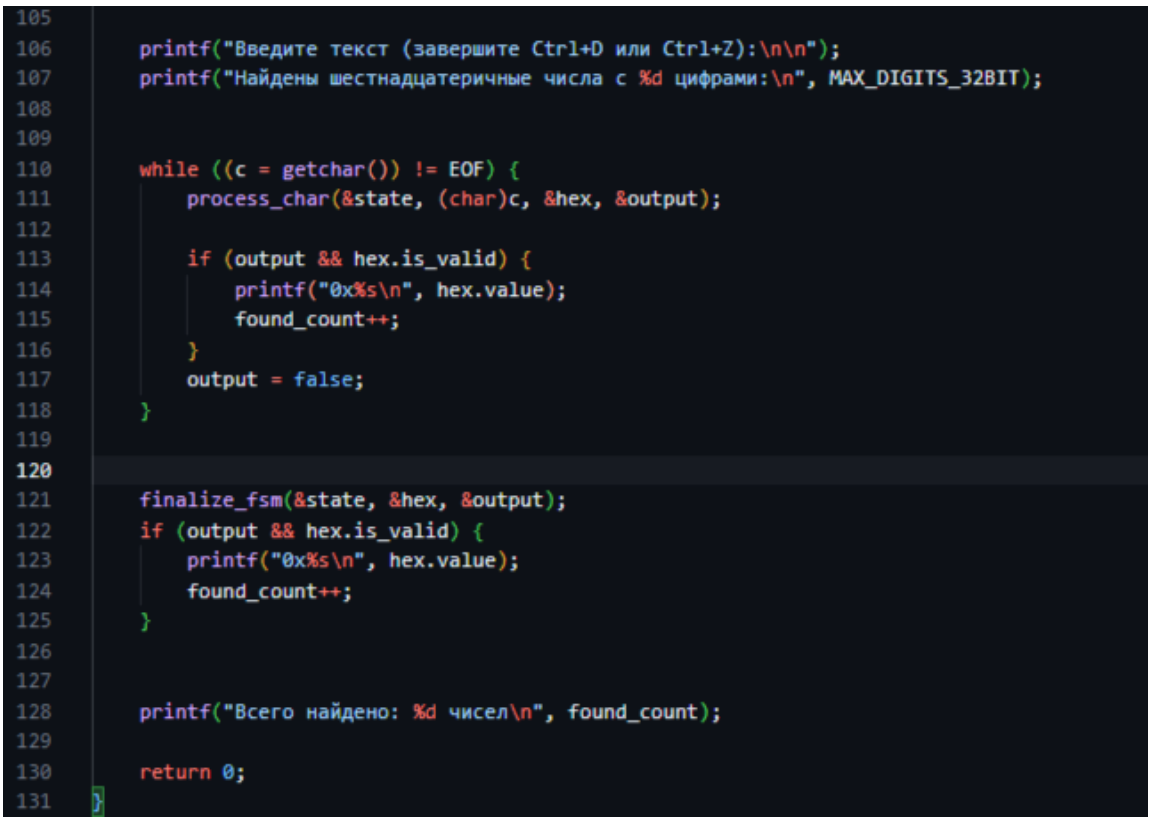
Ниже приведен код лабораторной работы номер 7.

```c
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

#define MAX_HEX_NUM 32
#define MAX_DIGITS_32BIT 8


typedef enum {
    STATE_START,
    STATE_ZERO,
    STATE_HEX_PREFIX,
    STATE_HEX_DIGIT,
    STATE_END
} FSMState;


typedef struct {
    bool is_valid;
    int digit_count;
    char value[MAX_HEX_NUM];
} HexNumber;


bool is_hex_digit(char c) {
    return isxdigit((unsigned char)c);
}


FSMState process_char(FSMState *state, char c, HexNumber *hex, bool *output) {
    *output = false;

    switch (*state) {
        case STATE_START:
            if (c == '0' || c == 'O') {
                *state = STATE_ZERO;
            }
            break;

        case STATE_ZERO:
            if (c == 'x' || c == 'X') {
                *state = STATE_HEX_PREFIX;
                hex->digit_count = 0;
                hex->value[0] = '\0';
            } else if (isdigit((unsigned char)c)) {
                *state = STATE_START;
            } else {
                *state = STATE_START;
            }
            break;

        case STATE_HEX_PREFIX:
            if (is_hex_digit(c)) {
                *state = STATE_HEX_DIGIT;
```

Рисунок Ж.1 — Начало кода 7 лабораторной работы.

```
                hex->value[hex->digit_count++] = c;
                hex->value[hex->digit_count] = '\0';
            } else {
                *state = STATE_START;
            }
            break;

        case STATE_HEX_DIGIT:
            if (is_hex_digit(c)) {
                if (hex->digit_count < MAX_HEX_NUM - 1) {
                    hex->value[hex->digit_count++] = c;
                    hex->value[hex->digit_count] = '\0';
                }
            } else {

                hex->is_valid = (hex->digit_count == MAX_DIGITS_32BIT);
                *output = true;
                *state = STATE_START;


                if (c == '0' || c == 'O') {
                    *state = STATE_ZERO;
                }
            }
            break;

        default:
            *state = STATE_START;
            break;
    }

    return *state;
}


void finalize_fsm(FSMState *state, HexNumber *hex, bool *output) {
    if (*state == STATE_HEX_DIGIT && hex->digit_count > 0) {
        hex->is_valid = (hex->digit_count == MAX_DIGITS_32BIT);
        *output = true;
    }
    *state = STATE_START;
}

int main(void) {
    FSMState state = STATE_START;
    HexNumber hex = {false, 0, ""};
    bool output = false;
    int c;
    int found_count = 0;
```

Рисунок Ж.2 — Продолжение кода 7 лабораторной работы.



```
105
106     printf("Введите текст (завершите Ctrl+D или Ctrl+Z):\n\n");
107     printf("Найдены шестнадцатеричные числа с %d цифрами:\n", MAX_DIGITS_32BIT);
108
109
110     while ((c = getchar()) != EOF) {
111         process_char(&state, (char)c, &hex, &output);
112
113         if (output && hex.is_valid) {
114             printf("0x%s\n", hex.value);
115             found_count++;
116         }
117         output = false;
118     }
119
120
121     finalize_fsm(&state, &hex, &output);
122     if (output && hex.is_valid) {
123         printf("0x%s\n", hex.value);
124         found_count++;
125     }
126
127
128     printf("Всего найдено: %d чисел\n", found_count);
129
130     return 0;
131 }
```

Рисунок Ж.3 — Продолжение кода 7 лабораторной работы.