МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационных систем и технологий
Специальность 6-05-0611-01 «Информационные системы и технологии»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Компьютерные языки разметки»
Тема: Веб-сайт «Калькулятор простых алгебрачиеских выражений»

Исполнитель
студент 1 курса 1 группы _____ И. А. Крупкевич
подпись, дата

Руководитель
ассистент _____ Д. В. Сазонова
должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите _____ _____
_____ _____
_____ _____
дата, подпись

Курсовой проект защищен с оценкой _____

Руководитель _____ _____ Д. В. Сазонова
подпись дата инициалы и фамилия

Минск, 2024

Содержание

1 Постановка задачи

1. Разработать веб-сайт с интерфейсом для ввода выражения и отображения результатов его вычисления.
2. Написать сервер на языке программирования C, который будет принимать запросы от веб-сайта, обрабатывать их и возвращать результаты вычислений.
3. Реализовать алгоритмы для обработки простых алгебраических выражений, включая операции сложения, вычитания, умножения и деления.
4. Обеспечить корректную работу веб-сайта и сервера, а также обработку возможных ошибок при вводе данных пользователем.

Сделать выводы о проделанной работе.

1.1 Обзор аналогичных решений

1.1.1 Wolframalpha

https://wolframalpha.com

WolframAlpha представляет собой мощный вычислительный инструмент, способный решать разнообразные математические задачи, предоставляя детальные ответы и графики.

1.1.2 Photomath

https://photomath.com/

В отличие от WolframAlpha, приложение Photomath предлагает уникальную функцию распознавания рукописных математических выражений с помощью камеры смартфона. Этот инновационный подход позволяет пользователям с легкостью вводить математические задачи, просто сфотографировав их, и получать мгновенные ответы. Photomath фокусируется на удобстве использования и доступности, делая процесс решения математических задач более интерактивным и интуитивным для пользователей всех уровней.

1.2 Техническое задание

1. Информация о проекте:
   - Количество веб-страниц: 3
   - Страницы и их содержание:
     - Первая страница: содержит большую кнопку ”Dive In”, которая переносит пользователя в удивительный мир математики.
     - Вторая страница: список теорем с возможностью просмотра их доказательств.
     - Третья страница: позволяет пользователю вычислить значение математического выражения.
2. Формирование требований к программному продукту:
   - Разработать веб-сайт с интуитивным интерфейсом и привлекательным дизайном.
   - Обеспечить быструю и удобную навигацию между страницами.
   - Создать страницу с перечнем теорем и возможностью просмотра их доказательств.
   - Разработать страницу для вычисления значений математических выражений с удобным интерфейсом для ввода данных.
3. Формулировка задач программного продукта:

- Создать страницу с перечнем теорем и возможностью просмотра их доказательств.
- Разработать функционал для вычисления значений математических выражений на отдельной странице.
- Обеспечить корректную работу всех элементов интерфейса и функций веб-сайта.
- Предоставить пользователю удобный и интерактивный опыт взаимодействия с математическими данными и вычислениями.

1.3 Выбор средств реализации программного продукта

При принятии решения о выборе средств для реализации программного продукта, я решил использовать среду разработки Visual Studio Code (VS Code). VS Code предоставляет удобную и гибкую среду для написания кода, обладает широким набором расширений и инструментов, что позволяет увеличить производительность и удобство разработки. Благодаря интеграции с различными языками программирования, отладчиками и системами контроля версий, VS Code обеспечивает эффективное взаимодействие с кодом и управление проектом. В результате использования VS Code я смог повысить эффективность разработки и улучшить качество программного продукта.

Дополнительно, для отображения математических формул на веб-сайте, я использовал библиотеку KaTeX. KaTeX обеспечивает быструю и качественную отрисовку математических выражений непосредственно в браузере, что позволяет представлять алгебраические формулы на веб-странице в удобном и профессиональном виде. Благодаря интеграции KaTeX в мой веб-сайт, пользователи могут легко вводить и просматривать математические выражения, что повышает удобство использования калькулятора и обеспечивает более качественный пользовательский опыт.

1.3.1 Вывод

Определен основной функционал программного продукта, включающий создание веб-сайта для математических вычислений с функцией перехода в интерактивный раздел, просмотром теорем и их доказательством, а также возможностью вычисления математических выражений. Для реализации проекта планируется использовать технологии KaTeX для отображения математических формул, язык программирования C для серверной части, а также HTML, CSS и JavaScript для создания пользовательского интерфейса веб-сайта. Этот стек технологий обеспечит удобство использования и функциональность программного продукта.

2 Проектирование страниц веб-сайта

2.1 Выбор способа верстки

При выборе способа верстки для моего проекта было решено использовать HTML и CSS в сочетании с библиотекой KaTeX. HTML (HyperText Markup breaklines=true,language) является основным языком разметки веб-страниц, который позволяет структурировать контент и создавать элементы интерфейса. CSS (Cascading Style Sheets) используется для стилизации веб-страниц, что позволяет задавать внешний вид элементов и макет страницы.

KaTeX представляет собой быструю и легковесную библиотеку для отрисовки математических выражений в формате TeX на веб-страницах. Благодаря своей эффективности и производительности, KaTeX обеспечивает быструю загрузку и отображение сложных математических формул без необходимости использования серверного рендеринга.

Использование HTML, CSS и библиотеки KaTeX в проекте позволило создать удобный и функциональный веб-интерфейс для взаимодействия с математическими данными, обеспечивая пользователю понятное и привлекательное представление информации.

2.2 Выбор стилевого оформления

При выборе стилевого оформления для моего проекта было принято использовать темную тему с белым текстом и минималистичным дизайном. Темный стиль интерфейса обеспечивает комфортное восприятие контента и снижает нагрузку на глаза пользователей, особенно при длительном использовании в условиях недостаточного освещения. Белый текст на темном фоне создает контрастный эффект, делая информацию более читаемой и выделяя ключевые элементы интерфейса.

Минималистичный дизайн был выбран для обеспечения простоты и ясности восприятия пользователем. Упрощенный стиль позволяет сосредоточить внимание на основном контенте и функционале веб-сайта, не отвлекая пользователя лишними деталями. Минимализм способствует лаконичности интерфейса, делая его более современным и эстетичным.

2.3 Выбор шрифтового оформления

Для шрифтового оформления я выбрал обычные шрифты. Основная цель была сделать текст читаемым и легко воспринимаемым. Обычные шрифты подходят для широкого круга пользователей и делают дизайн простым и доступным. Этот выбор помогает пользователям сосредоточиться на информации на сайте без лишних отвлечений.

2.4 Разработка логотипа

Figure 1: Логотип

При разработке логотипа была использована буква ”M”, стилизованная в виде числовых множеств, что символизирует связь с математикой. Название ”Mather” происходит от слова ”Math” - математика, что отражает основную тематику проекта.

2.5 Разработка пользовательских элементов

2.5.1 toggle switch

Разработка этого элемента интерфейса включала в себя создание ползунка, который плавно перемещается между двумя состояниями, отображая выбор пользователя. Я также добавил анимацию и эффекты перехода, чтобы сделать взаимодействие с переключателем более приятным и интуитивно понятным.

2.6 Разработка спецэффектов

Для создания красивой светящейся кнопки я использовал псевдоэлемент `::before`, который добавляет дополнительный слой перед содержимым кнопки. Я задал градиентный фон с помощью `linear-gradient`, установил анимацию для изменения позиции фона, добавил размытие с помощью `filter: blur()`, и настроил переход при наведении на кнопку для изменения размытия. Это создает эффект светящейся кнопки с плавными анимациями и стилизацией.

2.7 Выводы

1. Важность пользовательского опыта: Проектирование страниц веб-сайта играет ключевую роль в создании удобного и привлекательного пользовательского опыта. Грамотно спроектированные страницы способствуют легкому взаимодействию пользователя с сайтом и повышают его удовлетворенность.

2. Согласованность дизайна: Важно обеспечить согласованность дизайна на всех страницах веб-сайта. Это помогает создать цельное визуальное впечатление и узнаваемый стиль, что способствует узнаваемости бренда.

3. Адаптивность и отзывчивость: С учетом разнообразия устройств, на которых пользователи могут просматривать сайт, важно обеспечить адаптивность и отзывчивость дизайна. Это позволит сайту корректно отображаться на различных устройствах и улучшит общий пользовательский опыт.

4. Удобство навигации: Хорошо спроектированная навигация на страницах веб-сайта помогает пользователям быстро находить нужную информацию. Четкая структура и интуитивно понятные элементы навигации улучшают пользовательскую интеракцию.

5. Тестирование и оптимизация: Важно проводить тестирование дизайна страниц веб-сайта с целью выявления возможных проблем и улучшения пользовательского опыта. Оптимизация дизайна на основе обратной связи пользователей поможет создать более эффективный и привлекательный сайт.

В целом, проектирование страниц веб-сайта играет важную роль в создании успешного онлайн присутствия. Учитывая вышеперечисленные аспекты, можно создать сайт, который не только привлечет пользователей, но и обеспечит им приятный и удобный пользовательский опыт.

3 Реализация структуры веб-сайта

Представить листинги структуры веб-сайта. Обосновать выбор элементов в коде. Если это большой важный кусок кода, то покажите его в приложение в виде листинга, но укажите ссылку на приложение. Например, скрипт для создания таблиц представлен в приложении Б. В листингах интервалов после абзацев быть не должно.

3.1 Структура HTML-документа

3.2 Добавление таблиц стилей CSS

3.3 Использование стандартов XML (SVG)

3.4 Управление элементами DOM

3.5 Выводы

4 Тестирование веб-сайта

4.1 Адаптивный дизайн веб-сайта

- Проверка на различных устройствах: Веб-сайт был протестирован на различных устройствах с разными размерами экранов, чтобы убедиться, что дизайн адаптируется корректно и отображается правильно на всех устройствах.

- Тестирование в различных браузерах: Веб-сайт был протестирован в различных популярных браузерах, таких как Google Chrome, Mozilla Firefox, Safari и Microsoft Edge, чтобы убедиться, что он отображается одинаково хорошо во всех браузерах.

- Использование инструментов разработчика: Для более детального анализа адаптивности веб-сайта были использованы инструменты разработчика браузера, позволяющие эмулировать различные устройства и размеры экранов.

После проведения тестирования были выявлены и исправлены возможные проблемы с адаптивностью веб-сайта, что позволило обеспечить оптимальное пользовательское взаимодействие на всех устройствах и в различных браузерах.

4.2 Кроссбраузерность веб-сайта

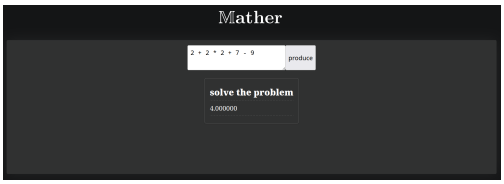

Figure 2: страница с калькулятором на Chrome

Figure 3: начальная страница на Chrome

Figure 4: страница с теоремами на Chrome

Figure 5: страница с калькулятором на Firefox

Figure 6: начальная страница на Firefox

Figure 7: страница с теоремами на Firefox

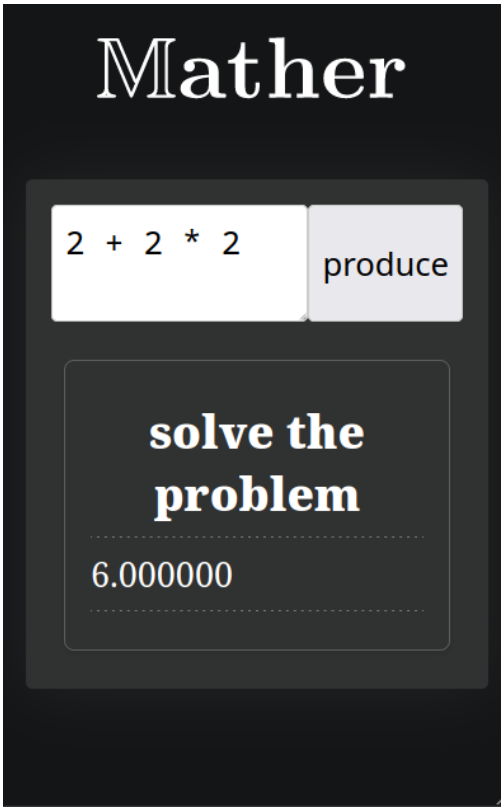

Figure 8: страница с калькулятором на Телефоне

Figure 9: начальная страница на Телефоне

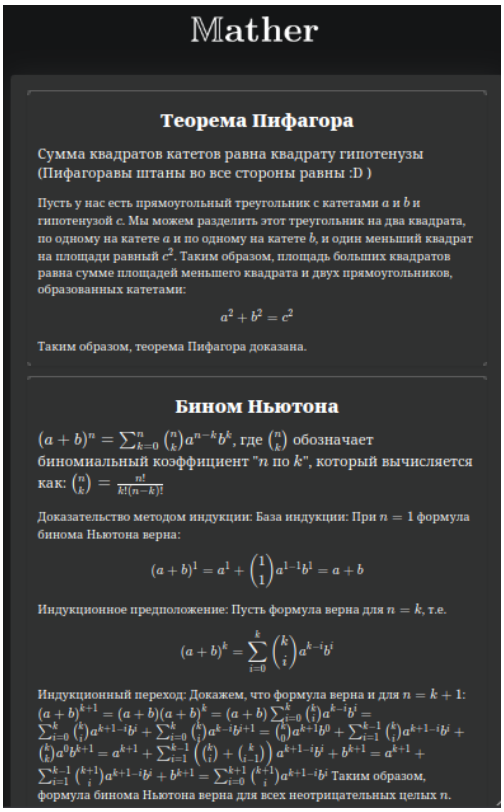

Figure 10: страница с теоремами на Телефоне

4.3 Руководство пользователя

Figure 11: начальная страница

На начальной странице представлен удобный интерфейс, позволяющий пользователям легко навигироваться по различным разделам. Нажав на кнопку ”Dive In”, пользователь перемещается на страницу, где представлены различные теоремы. В то же время, активация переключателя низу экрана направляет пользователя на страницу с калькулятором. Здесь пользователи могут вводить математические выражения и получать соответствующие ответы. Такой подход обеспечивает удобство использования и эффективный доступ к различным функциям веб-ресурса.

4.4 Выводы

В ходе выполнения курсового проекта был разработан веб-сайт с удобным интерфейсом, позволяющим пользователям изучать теоремы и использовать калькулятор для математических вычислений. Дополнительно к использованию стандартных технологий, был написан сервер на языке программирования C для обеспечения более эффективной обработки запросов и управления данными.

Ссылка на GitHub: https://github.com/stoopotec/Counter

5 Приложение А: листинг файла ./src/http.c

```
#include <unistd.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

#include "server.h"
#include "internet.h"
#include "http.h"
#include "messages.h"


#define CHUNK_SIZE 1024
// #define RECEIVE_DELAY


int strcmp_start(const char* s1, const char* s2) {
    for (size_t i = 0; s1[i] != '\0' && s2[i] != '\0'; ++i)
        if (s1[i] != s2[i]) return 0;
    return 1;
}

enum METHOD_E get_method(char* http_header) {
    if (strcmp_start(http_header, TOSTRING(GET) )) return GET;
    if (strcmp_start(http_header, TOSTRING(POST))) return POST;
    if (strcmp_start(http_header, TOSTRING(CALC))) return CALC;
    return UNDEFINED_METHOD;
}




char* strcpy_a(const char* url, const char* add_to_start) {
    LIST(char) path = {0};
    if (add_to_start != NULL) {
        for (; *add_to_start != '\0'; ++add_to_start) {
            LIST_APPEND(*add_to_start, path);
        }
    }
    for (; *url != '?' && *url != '\0'; ++url) {
        LIST_APPEND(*url, path);
    }
    LIST_APPEND('\0', path);
    return path.data;
}


list_url_arg_t get_url_args(const char* url) {
    list_url_arg_t args = {0};
    for (; *url != '?' && *url != '\0'; ++url) {}
    if (*url == '\0') return args;

    url += 1;
```

```
    url_arg_t null_arg = {.key = NULL, .key_len = 0, .val = NULL, .val_len = 0};
    LIST_APPEND(null_arg, args);

    args.data[args.length-1].key = url;


    for (size_t i = 0; url[i] != '\0'; ++i) {
        if (url[i] == '&') {
            LIST_APPEND(null_arg, args);
            args.data[args.length-2].val_len = url + i - args.data[args.length-2].
                val;
            args.data[args.length-1].key = url + i + 1;
            continue;
        }
        if (url[i] == '=') {
            args.data[args.length-1].key_len = url + i - args.data[args.length-1].
                key;
            args.data[args.length-1].val = url + i + 1;
            continue;
        }
        if (url[i+1] == '\0') {
            args.data[args.length-1].val_len = url + i+1 - args.data[args.length
                -1].val;
        }
    }

    return args;
}


struct http_message server_response_custom(enum HTTP_CODES_E status_code, char*
    body) {
    return (struct http_message) {
        .http_version = "HTTP/1.1",
        .status_code = {
            '0' + (status_code % 1000) / 100,
            '0' + (status_code % 100) / 10,
            '0' + (status_code % 10) / 1,
            '\0',
        },
        .status_description = get_code_comment(status_code),
        .method = NULL,
        .url = NULL,
        .url_args = {0},
        .headers = {0},
        .body = body,
    };
}

struct http_message server_response_default() {
    return (struct http_message) {
        .http_version = "HTTP/1.1",
        .status_code = "200",
        .status_description = get_code_comment(CODE_200_OK),
        .method = NULL,
```

```
        .url = NULL,
        .url_args = {0},
        .headers = {0},
        .body = NULL,
    };
}

struct http_message server_response_notfound() {
    return (struct http_message) {
        .http_version = "HTTP/1.1",
        .status_code = "404",
        .status_description = get_code_comment(CODE_404_NOT_FOUND),
        .method = NULL,
        .url = NULL,
        .url_args = {0},
        .headers = {0},
        .body =
"␣␣␣␣␣␣␣␣<!DOCTYPE␣html>\n"
"␣␣␣␣␣␣␣␣<html␣lang=\"en\">\n"
"␣␣␣␣␣␣␣␣<head>\n"
"␣␣␣␣␣␣␣␣␣␣␣␣<meta␣charset=\"UTF-8\">\n"
"␣␣␣␣␣␣␣␣␣␣␣␣<meta␣name=\"viewport\"␣content=\"width=device-width,␣initial-scale
    =1.0\">\n"
"␣␣␣␣␣␣␣␣␣␣␣␣<title>not␣found␣:(</title>\n"
"␣␣␣␣␣␣␣␣</head>\n"
"␣␣␣␣␣␣␣␣␣␣␣␣<body>\n"
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<div␣style=\"width:␣100vw;␣height:␣100vh;␣text-align:␣center;\">\
    n"
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<h1>requested␣page␣not␣found</h1>\n"
"␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣</div>\n"
"␣␣␣␣␣␣␣␣␣␣␣␣</body>\n"
"␣␣␣␣␣␣␣␣</html>\n",
    };
}

void send_http_message(int sockfd, struct http_message* message) {

    send(sockfd, message->http_version, 7, MSG_MORE);
    send(sockfd, "␣", 1, MSG_MORE);
    send(sockfd, message->http_version, 3, MSG_MORE);
    send(sockfd, "␣", 1, MSG_MORE);
    send(sockfd, message->http_version, strlen(message->http_version), MSG_MORE);
    send(sockfd, "\n", 1, MSG_MORE);

    for (size_t i = 0; i < message->headers.length; ++i) {
        send(sockfd, message->headers.data[i].key, 1, MSG_MORE);
        send(sockfd, ":␣", 2, MSG_MORE);
        send(sockfd, message->headers.data[i].value, 1, MSG_MORE);
        send(sockfd, "\n", 1, MSG_MORE);
    }
    send(sockfd, "\n", 1, MSG_MORE);

    send(sockfd, message->body, strlen(message->body), MSG_WAITALL);
}
```

```
void next_word_and(char* text, char** word, size_t* word_len, const char*
    separators) {
    *word = text;
    *word_len = 0;
    for (size_t i = 0; text[i] != '\0'; ++i) {
        *word_len = i;
        for (size_t s = 0; separators[s] != '\0'; ++s) {
            if (text[i+s] != separators[s]) {
                goto ce;
            }
        }
        return;
        ce:
    }
    *word_len += 1;
    return;
}

// 's' in 'ors' stands for "skip first separators if there are any"
void next_word_ands(char* text, char** word, size_t* word_len, const char*
    separators) {

    for (; *text != '\0'; ++text) {
        for (size_t s = 0; separators[s] != '\0'; ++s) {
            if (separators[s] == *text) goto te;
        }
        break;
        te:
    }

    return next_word_and(text, word, word_len, separators);
}


void next_word_or(char* text, char** word, size_t* word_len, const char*
    separators) {
    *word = text;
    *word_len = 0;
    for (size_t i = 0; text[i] != '\0'; ++i) {
        *word_len = i;
        for (size_t s = 0; separators[s] != '\0'; ++s) {
            if (text[i] == separators[s]) {
                return;
            }
        }
    }
    *word_len += 1;
    return;
}

// 's' in 'ors' stands for "skip first separators if there are any"
void next_word_ors(char* text, char** word, size_t* word_len, const char*
    separators) {

    for (; *text != '\0'; ++text) {
        for (size_t s = 0; separators[s] != '\0'; ++s) {
```

```
            if (separators[s] == *text) goto te;
        }
        break;
        te:
    }

    return next_word_or(text, word, word_len, separators);
}

void sget_http_message(char* text, struct http_message* message) { // WARN:  ,
            text     (        ),

    size_t word_size;

    next_word_ors(text, &message->method, &word_size, "␣");
    message->method[word_size] = '\0';
    text = message->method + word_size + 1;

    next_word_ors(text, &message->url, &word_size, "␣");
    message->url[word_size] = '\0';
    text = message->url + word_size + 1;

    do {
        char* url = message->url;
        for (; *url != '?' && *url != '\0'; ++url) {}
        if (*url == '\0') break;


        for (size_t i = 0; url[i] != '\0'; ++i) {
            if (url[i] == '&' || url[i] == '?') {
                url[i] = '\0';
                if (url[i+1] != '\0') {
                    LIST_EXPAND_BY(message->url_args, 1);
                    message->url_args.data[message->url_args.length-1].key = url +
                         i + 1;
                    message->url_args.data[message->url_args.length-1].value =
                        NULL;
                }
                continue;
            }
            if (url[i] == '=') {
                url[i] = '\0';
                if (url[i+1] != '\0') {
                    message->url_args.data[message->url_args.length-1].value = url
                         + i + 1;
                }
                continue;
            }
        }

    } while (0);

    next_word_ors(text, &message->http_version, &word_size, "␣");
    message->http_version[word_size] = '\0';
    text = message->http_version + word_size + 2;
```

```
    for (; *text != '\r' && *text != '\0';) {
        LIST_EXPAND_BY(message->headers, 1);
        next_word_ors(text, &message->headers.data[message->headers.length].key, &
            word_size, ":");
        message->headers.data[message->headers.length].key[word_size] = '\0';
        text = message->headers.data[message->headers.length].key + word_size + 2;

        next_word_ors(text, &message->headers.data[message->headers.length].value,
             &word_size, "\r");
        message->headers.data[message->headers.length].value[word_size] = '\0';
        text = message->headers.data[message->headers.length].value + word_size +
            2;

        message->headers.length += 1;
    }

    message->body = text + 2;

    return;
}

http_message_additional get_http_message(int sockfd) {

    printf(INFO E_FOREGROUND_RED "get_http_message\n" E_RESET);
    http_message_additional r = {0};

    #if defined(RECEIVE_DELAY)
        struct timespec ts;
        ts.tv_sec = 2;
        ts.tv_nsec = 100000000;
        printf("   ␣    \n");
        nanosleep(&ts, NULL);
        printf("   ␣    \n");
    #endif

    size_t data_size = 0;
    {
        size_t buffer_size = 0;
        ssize_t bytes_received;

        do {
            buffer_size += CHUNK_SIZE;
            r.alloced_raw_content = realloc(r.alloced_raw_content, buffer_size);
            if (r.alloced_raw_content == NULL) {
                perror(ERR "buy␣more␣ram,␣lol\n");
                exit(EXIT_FAILURE);
            }

            bytes_received = recv(sockfd, r.alloced_raw_content + data_size,
                CHUNK_SIZE, 0);
            if (bytes_received < 0) {
                perror(ERR "Error␣receiving␣data␣from␣socket\n");
                exit(EXIT_FAILURE);
            }
```

```
            data_size += bytes_received;
        } while (bytes_received == CHUNK_SIZE);

        // Resize buffer to actual data size
        r.alloced_raw_content = realloc(r.alloced_raw_content, data_size + 1);
        if (r.alloced_raw_content == NULL) {
            perror(ERR "buy more ram, lol\n");
            exit(EXIT_FAILURE);
        }
        r.alloced_raw_content[data_size] = '\0'; // Null-terminate the string
    }

    printf(INFO "raw content:\n" E_ITALIC "%s" E_RESET "\n", r.alloced_raw_content
        );

    sget_http_message(r.alloced_raw_content, &r.message);

    return r;
}


const char* get_code_comment(enum HTTP_CODES_E code) {

    switch (code) {

        case 100: return "Continue";
        case 101: return "Switching Protocols";
        case 102: return "Processing";
        case 103: return "Early Hints";

        case 200: return "OK";
        case 201: return "Created";
        case 202: return "Accepted";
        case 203: return "Non-Authoritative Information";
        case 204: return "No Content";
        case 205: return "Reset Content";
        case 206: return "Partial Content";
        case 207: return "Multi-Status";
        case 208: return "Already Reported";
        case 226: return "IM Used";

        case 300: return "Multiple Choices";
        case 301: return "Moved Permanently";
        case 302: return "Found";
        case 303: return "See Other";
        case 304: return "Not Modified";
        case 305: return "Use Proxy";
        case 306: return "unused";
        case 307: return "Temporary Redirect";
        case 308: return "Permanent Redirect";

        case 400: return "Bad Request";
        case 401: return "Unauthorized";
        case 402: return "Payment Required";
        case 403: return "Forbidden";
        case 404: return "Not Found";
```

```
        case 405: return "Method Not Allowed";
        case 406: return "Not Acceptable";
        case 407: return "Proxy Authentication Required";
        case 408: return "Request Timeout";
        case 409: return "Conflict";
        case 410: return "Gone";
        case 411: return "Length Required";
        case 412: return "Precondition Failed";
        case 413: return "Payload Too Large";
        case 414: return "URI Too Long";
        case 415: return "Unsupported Media Type";
        case 416: return "Range Not Satisfiable";
        case 417: return "Expectation Failed";
        case 418: return "I\'m a teapot";
        case 421: return "Misdirected Request";
        case 422: return "Unprocessable Content";
        case 423: return "Locked";
        case 424: return "Failed Dependency";
        case 425: return "Too Early";
        case 426: return "Upgrade Required";
        case 428: return "Precondition Required";
        case 429: return "Too Many Requests";
        case 431: return "Request Header Fields Too Large";
        case 451: return "Unavailable For Legal Reasons";

        case 500: return "Internal Server Error";
        case 501: return "Not Implemented";
        case 502: return "Bad Gateway";
        case 503: return "Service Unavailable";
        case 504: return "Gateway Timeout";
        case 505: return "HTTP Version Not Supported";
        case 506: return "Variant Also Negotiates";
        case 507: return "Insufficient Storage";
        case 508: return "Loop Detected";
        case 510: return "Not Extended";
        case 511: return "Network Authentication Required";

    }

}
```

6 Приложение Б: листинг файла ./src/http.h

```
#pragma once

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

#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>
```

```
#include "list.h"



enum METHOD_E {
    UNDEFINED_METHOD = 0,
    GET  = 1,
    POST = 2,
    CALC = 3,
};

enum METHOD_E get_method(char* http_header);

const char* define_content_type(const char* filename);


typedef struct url_arg {
    const char* key;
    size_t key_len;
    const char* val;
    size_t val_len;
} url_arg_t;


typedef LIST(url_arg_t) list_url_arg_t;
char* get_url_alloc(char* http);

list_url_arg_t get_url_args(const char* url);

char* strcpy_a(const char* url, const char* add_to_start);









enum HTTP_CODES_E {

//

    //          ,       ,                                   ,              .
    CODE_100_CONTINUE = 100,

    //                              Upgrade                 ,                 .
    CODE_101_SWITCHING_PROTOCOLS = 101,
```

```
    //                    ,          .
    CODE_102_PROCESSING = 102,

    //                                        Link,
                                                                          ,
                    .
    CODE_103_EARLY_HINTS = 103,


//

    //             .              «»            HTTP:
    //     GET:                          .
    //     HEAD:              ,             .
    //     PUT   POST:    ,               ,              .
    //     TRACE:                        ,          .
    CODE_200_OK = 200,

    //             ,                        .          ,              POST   PUT.
    CODE_201_CREATED = 201,

    //        ,            .        «»    ,        HTTP
                                                  .                  ,
                                        ,                 .
    CODE_202_ACCEPTED = 202,

    //                                ,                    ,                 .
                                                 .
        200 OK.
    CODE_203_NON_AUTHORITATIVE_INFORMATION = 203,

    //                          ,                     .
                                                        ,                  .
    CODE_204_NO_CONTENT = 204,

    //                  ,                          ,              .
    CODE_205_RESET_CONTENT = 205,

    //                ,                          Range                    .
    CODE_206_PARTIAL_CONTENT = 206,

    //                              ,                                .
    CODE_207_MULTI_STATUS = 207,

    //                     <dav:propstat>,                              «»            .
    CODE_208_ALREADY_REPORTED = 208,

    //                     GET        ,                                          (
        «»).
    CODE_226_IM_USED = 226,


//

    //                       .                                    .
                                          ,                HTML   -           ,
```

```
                          .
    CODE_300_MULTIPLE_CHOICES = 300,

    // URL   -                        .     URL   -          .
    CODE_301_MOVED_PERMANENTLY = 301,

    // URI                           .                                URI.        ,
       URI                               .
    CODE_302_FOUND = 302,

    //                                URI          GET.
    CODE_303_SEE_OTHER = 303,

    //                         .           ,               ,
                                                 .
    CODE_304_NOT_MODIFIED = 304,

    //                               .                               HTTP
                     -           ,                             -.
    CODE_305_USE_PROXY = 305, // depricated

    //                  .                            HTTP/1.1.
    CODE_306_UNUSED = 306,

    //                                URI          ,                        .
                    ,            302 Found,            ,
                                          :                       POST,   POST
                         .
    CODE_307_TEMPORARY_REDIRECT = 307,

    //                     URI,                  Location.                ,
                   301 Moved Permanently,           ,
                                          :                       POST,   POST
                         .
    CODE_308_PERMANENT_REDIRECT = 308,


//

    //                                  -    -,                           (,
       ,                   ).
    CODE_400_BAD_REQUEST = 400,

    //         HTTP                         «»,                          «».       ,
                          ,                 .
    CODE_401_UNAUTHORIZED = 401,

    //                                   .
                                                               ,
                                                 .
    CODE_402_PAYMENT_REQUIRED = 402, // experemental

    //                          ,               ,                             .
                401 Unauthorized,             .
    CODE_403_FORBIDDEN = 403,
```

```
// . , URL - . API ,
, . 403 Forbidden,
. - .
CODE_404_NOT_FOUND = 404,

// , . , API DELETE
.
CODE_405_METHOD_NOT_ALLOWED = 405,

// , , .
CODE_406_NOT_ACCEPTABLE = 406,

// 401 Unauthorized, -.
CODE_407_PROXY_AUTHENTICATION_REQUIRED = 407,

// , . ,
. , (
Chrome, Firefox 27+ IE9) HTTP.
, .
CODE_408_REQUEST_TIMEOUT = 408,

// .
CODE_409_CONFLICT = 409,

// , . ,
. HTTP ,
« ». API ,
, .
CODE_410_GONE = 410,

// , Content-Length, .
CODE_411_LENGTH_REQUIRED = 411,

// , .
CODE_412_PRECONDITION_FAILED = 412,

// , . Retry-
After.
CODE_413_PAYLOAD_TOO_LARGE = 413,

// URI , .
CODE_414_URI_TOO_LONG = 414,

// , .
CODE_415_UNSUPPORTED_MEDIA_TYPE = 415,

// , Range.
CODE_416_RANGE_NOT_SATISFIABLE = 416,

// , Expect.
CODE_417_EXPECTATION_FAILED = 417,

// «» : .
CODE_418_IM_A_TEAPOT = 418,

// . , URI .
```

```
    CODE_421_MISDIRECTED_REQUEST = 421,

    //          ,          -          .
    CODE_422_UNPROCESSABLE_CONTENT = 422,

    //          .
    CODE_423_LOCKED = 423,

    //          -          .
    CODE_424_FAILED_DEPENDENCY = 424,

    //          ,          .
    CODE_425_TOO_EARLY = 425, // experemental

    //          ,
                                                Upgrade.
    CODE_426_UPGRADE_REQUIRED = 426,

    //     ,          .
                                                     .
    CODE_428_PRECONDITION_REQUIRED = 428,

    //                    .
    CODE_429_TOO_MANY_REQUESTS = 429,

    //          ,          .
                                 .
    CODE_431_REQUEST_HEADER_FIELDS_TOO_LARGE = 431,

    //          ,          ,          -,
                   .
    CODE_451_UNAVAILABLE_FOR_LEGAL_REASONS = 451,

//

    //          ,          .
    CODE_500_INTERNAL_SERVER_ERROR = 500,

    //                    .    GET   HEAD
                                 .
    CODE_501_NOT_IMPLEMENTED = 501,

    //          ,          ,
                                 .
    CODE_502_BAD_GATEWAY = 502,

    //          .                              .
                                    ,      HTTP     - Retry
       -After          .    ,                    ,
                   .
    CODE_503_SERVICE_UNAVAILABLE = 503,

    //          ,          ,
                                      .
    CODE_504_GATEWAY_TIMEOUT = 504,
```

```
    //                  HTTP                .
    CODE_505_HTTP_VERSION_NOT_SUPPORTED = 505,

    //                              :                                          .
    CODE_506_VARIANT_ALSO_NEGOTIATES = 506,

    //          ,                               .
    CODE_507_INSUFFICIENT_STORAGE = 507,

    //          ,                                          .
    CODE_508_LOOP_DETECTED = 508,

    //                            .
    CODE_510_NOT_EXTENDED = 510,

    //                                        .
    CODE_511_NETWORK_AUTHENTICATION_REQUIRED = 511,


};

const char* get_code_comment(enum HTTP_CODES_E code);




struct http_message {
    char* http_version;
    char  status_code[4];
    char* status_description;
    char* method;
    char* url;
    LIST_ANON(struct {char* key; char* value;}) headers;
    LIST_ANON(struct {char* key; char* value;}) url_args;
    char* body;
};

struct http_message server_response_custom(enum HTTP_CODES_E status_code, char*
    body);
struct http_message server_response_default();
struct http_message server_response_notfound();



void send_http_message(int sockfd, struct http_message* message);

typedef struct {struct http_message message; char* alloced_raw_content;}
    http_message_additional;
http_message_additional get_http_message(int sockfd);
```

7 Приложение B: листинг файла ./src/internet.c

```
#include "internet.h"



struct sockaddr create_sockaddr(socklen_t* lengthr, uint16_t port) {

    struct sockaddr_in address;
    *lengthr = sizeof(address);
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(port);

    return *(struct sockaddr*)&address;

}


int create_server_socket(struct sockaddr* address) {
    int server_socket;

    if ((server_socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("ERROR: socket failed\n");
        return server_socket;
    }

    int reuse = 1;
    if (setsockopt(server_socket, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof(reuse))
        < 0) {
        perror("ERROR:          SO_REUSEADDR\n");
        return -1;
    }

    printf("INFO: binding socket to INADDR_ANY\n");


    if (bind(server_socket, address, sizeof(*address)) < 0) {
        perror("ERROR: bind failed\n");
        return -1;
    }
    printf("INFO: done!\n");


    printf("INFO: initialization of listen...\n");

    if (listen(server_socket, 5) < 0) {
        perror("ERROR: listen failed\n");
        return -1;
    }

    printf("INFO: done!\n");

    return server_socket;
}
```

8 Приложение Г: листинг файла ./src/internet.h

```
#pragma once

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

#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>

#include "list.h"



struct sockaddr create_sockaddr(socklen_t* lengthr, uint16_t port);

int create_server_socket(struct sockaddr* address);

long find(const char* from, const char* what);
```

9 Приложение Д: листинг файла ./src/list.h

```
#pragma once

#include <string.h>


#ifndef LIST_MALLOC_
    #include <stdlib.h>
    #define LIST_MALLOC_ malloc
#endif

#ifndef LIST_CALLOC_
    #include <stdlib.h>
    #define LIST_CALLOC_ calloc
#endif

#ifndef LIST_REALLOC_
    #include <stdlib.h>
    #define LIST_REALLOC_ realloc
#endif

#ifndef LIST_FREE_
    #include <stdlib.h>
    #define LIST_FREE_ free
#endif


#define APPEND_ALLOC_LENGTH ((size_t)10)
```

```
#define MAX(a, b) ((a) < (b) ? (b) : (a))
#define MIN(a, b) ((a) < (b) ? (a) : (b))

#define SWAP(a, b, data_type) {\
    data_type x = a;\
    a = b;\
    b = x;\
}\


#define LIST_FREE(list) (LIST_FREE_((list).data))


#define LIST_EXPAND_BY(list, expand_by) {\
    list.alloc_length += expand_by;\
    list.data = LIST_REALLOC_((list.data), list.alloc_length * sizeof(*(list.data)
        ));\
}\

#define LIST_EXPAND_TO(list, expand_to) {\
    list.alloc_length = MAX(list.alloc_length, expand_to), \
    list.data = LIST_REALLOC_((list.data), list.alloc_length * sizeof(*(list.data)
        ));\
}\

#define LIST_INIT(list) {\
    list.alloc_length = LIST_ALLOC_LENGTH;\
    list.length = 0;\
    list.data = LIST_CALLOC_(list.alloc_length, sizeof(*list.data));\
}\

#define LIST_INIT_LEN(list, alloc_len) {\
    list.alloc_length = alloc_len;\
    list.length = 0;\
    list.data = LIST_CALLOC_(list.alloc_length, sizeof(*list.data));\
}\


#define LIST_APPEND(x, list) {\
    if (list.data == NULL) {\
        list.alloc_length = APPEND_ALLOC_LENGTH;\
        list.length = 0;\
        list.data = LIST_MALLOC_(list.alloc_length * sizeof(*(list.data)));\
    }\
    if (list.length >= list.alloc_length) {\
        LIST_EXPAND_BY(list, APPEND_ALLOC_LENGTH)\
    }\
    list.data[list.length++] = x;\
}

#define LIST_APPEND_RANGE(x, x_length, list) if (x != NULL) {\
    if (list.data == NULL) {\
        list.alloc_length = MAX(APPEND_ALLOC_LENGTH, x_length);\
        list.length = 0;\
        list.data = LIST_MALLOC_(list.alloc_length * sizeof(*(list.data)));\
    }\
```

```
    if (list.length + x_length - 1 >= list.alloc_length) {\
        LIST_EXPAND_BY(list, APPEND_ALLOC_LENGTH + x_length)\
    }\
    memcpy(list.data + list.length, x, x_length * sizeof(*list.data));\
    list.length += x_length;\
}\


#define LIST_PUSH_AT(x, list, index) {\
    if (list.data == NULL) {\
        list.alloc_length = APPEND_ALLOC_LENGTH;\
        list.length = 0;\
        list.data = LIST_MALLOC_(list.alloc_length * sizeof(*(list.data)));\
    }\
    if (index < list.length) {\
        LIST_APPEND(list.data[list.length - 1], list)\
        for (int i = list.length-2; i >= index && i < list.length; --i)\
            list.data[i+1] = list.data[i];\
        list.data[index] = x;\
    }\
    else LIST_APPEND(x, list)\
}

#define LIST_PUSH_RANGE_AT(x, x_length, list, index) if (x != NULL) {\
    if (list.data == NULL) {\
        list.alloc_length = APPEND_ALLOC_LENGTH;\
        list.length = 0;\
        list.data = LIST_MALLOC_(list.alloc_length * sizeof(*(list.data)));\
    }\
    if (index < list.length) {\
        if (index + x_length >= list.alloc_length || 1) {\
            LIST_EXPAND_BY(list, x_length)\
        }\
        for (int i = list.length - 1; i >= index; --i)\
            list.data[i+x_length] = list.data[i];\
        list.length += x_length;\
        memcpy(list.data + index, x, x_length * sizeof(*list.data));\
    }\
    else LIST_APPEND_RANGE(x, x_length, list)\
}\

#define LIST(type) struct list_##type {\
    type* data;\
    size_t length;\
    size_t alloc_length;\
}

#define LIST_ANON(type) struct {\
    type* data;\
    size_t length;\
    size_t alloc_length;\
}
```

10 Приложение E: листинг файла ./src/main.c

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

#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>

#include <signal.h>

#include "internet.h"
#include "server.h"
#include "messages.h"


#define PORT 8080


int socketfd;

void inter(int integer) {
    printf(INFO "closing␣the␣server␣socket...\n");
    close(socketfd);
    exit(EXIT_SUCCESS);
}


int main() {
    socketfd = socket(AF_INET, SOCK_STREAM, 0);
    if (-1 == socketfd) {
        fprintf(stderr, ERR "cannot␣create␣socket\n");
        exit(EXIT_FAILURE);
    }


    signal(SIGINT, (__sighandler_t)&inter);


    struct sockaddr_in addr_in;
    memset(&addr_in, 0, sizeof(addr_in));
    addr_in.sin_family = AF_INET;
    addr_in.sin_port = htons(PORT);
    addr_in.sin_addr.s_addr = INADDR_ANY;


    if (-1 == bind(socketfd, (struct sockaddr*)&addr_in, sizeof(addr_in))) {
        fprintf(stderr, ERR "cannot␣bind\n");
        exit(EXIT_FAILURE);
    }

    if (-1 == listen(socketfd, 10)) {
        fprintf(stderr, ERR "cannot␣listen\n");
```

```
        exit(EXIT_FAILURE);
    }

    printf(INFO "server␣running!␣localhost:␣" E_ITALIC E_UNDERLINE "http://
        localhost:%d/index.html" E_RESET "\n", PORT);

    while (1)
        serve_client(accept(socketfd, NULL, NULL));


    close(socketfd);



    return 0;
}
```

11 Приложение Ж: листинг файла ./src/messages.h

```
#pragma once

#define TOSTRING(x) #x

#define E_RESET      "\033[0m"
#define E_BOLD       "\033[1m"
// #define E_DIM        "\033[2m"
#define E_ITALIC     "\033[3m"
#define E_UNDERLINE  "\033[4m"
#define E_BLINK      "\033[5m"
// #define E_RAPID_BLINK      "\033[6m"
#define E_INVERT     "\033[7m"
#define E_HIDE       "\033[8m"
#define E_CROSSED    "\033[9m"
#define E_ITALIC     "\033[3m"


#define E_FOREGROUND_RGB(r, g, b) "\033[38;2;"#r";"#g";"#b"m"
#define E_BACKGROUND_RGB(r, g, b) "\033[48;2;"#r";"#g";"#b"m"

#define E_FOREGROUND_BLACK "\033[30m"
#define E_BACKGROUND_BLACK "\033[40m"

#define E_FOREGROUND_RED "\033[31m"
#define E_BACKGROUND_RED "\033[41m"

#define E_FOREGROUND_GREEN "\033[32m"
#define E_BACKGROUND_GREEN "\033[42m"

#define E_FOREGROUND_YELLOW "\033[33m"
#define E_BACKGROUND_YELLOW "\033[43m"

#define E_FOREGROUND_BLUE "\033[34m"
#define E_BACKGROUND_BLUE "\033[44m"
```

```
#define E_FOREGROUND_MAGENTA "\033[35m"
#define E_BACKGROUND_MAGENTA "\033[45m"

#define E_FOREGROUND_CYAN "\033[36m"
#define E_BACKGROUND_CYAN "\033[46m"

#define E_FOREGROUND_WHITE "\033[37m"
#define E_BACKGROUND_WHITE "\033[47m"

#define E_FOREGROUND_GRAY "\033[38;5;90m"
#define E_BACKGROUND_GRAY "\033[48;5;100m"

#define E_FOREGROUND_BRIGHT_RED "\033[38;5;91m"
#define E_BACKGROUND_BRIGHT_RED "\033[48;5;101m"

#define E_FOREGROUND_BRIGHT_GREEN "\033[38;5;92m"
#define E_BACKGROUND_BRIGHT_GREEN "\033[48;5;102m"

#define E_FOREGROUND_BRIGHT_YELLOW "\033[38;5;93m"
#define E_BACKGROUND_BRIGHT_YELLOW "\033[48;5;103m"

#define E_FOREGROUND_BRIGHT_BLUE "\033[38;5;94m"
#define E_BACKGROUND_BRIGHT_BLUE "\033[48;5;104m"

#define E_FOREGROUND_BRIGHT_MAGENTA "\033[38;5;95m"
#define E_BACKGROUND_BRIGHT_MAGENTA "\033[48;5;105m"

#define E_FOREGROUND_BRIGHT_CYAN "\033[38;5;96m"
#define E_BACKGROUND_BRIGHT_CYAN "\033[48;5;106m"

#define E_FOREGROUND_BRIGHT_WHITE "\033[38;5;97m"
#define E_BACKGROUND_BRIGHT_WHITE "\033[48;5;107m"


#define INFO E_FOREGROUND_CYAN "INFO:␣" E_RESET
#define ERR E_FOREGROUND_RED "ERR:␣␣" E_RESET
#define WARN E_FOREGROUND_YELLOW "WARN:␣" E_RESET
```

12 Приложение И: листинг файла ./src/send_files.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

#include "server.h"
#include "internet.h"
#include "messages.h"
#include "http.h"


const char* define_content_type(const char* filename);


int send_small_file(int socketfd, const char* filename) {
```

```
    if (filename == NULL) {
        printf(ERR "cannot␣send␣file.␣" E_ITALIC TOSTRING(filename) E_RESET"␣in␣
            NULL\n");
        return EXIT_FAILURE;
    }
    if (socketfd <= 0) {
        printf(ERR "cannot␣send␣file␣" E_ITALIC "%s" E_RESET "\n␣becaurse␣of␣
            corrupted␣socketfd\n", filename);
        return EXIT_FAILURE;
    }



    FILE* file = fopen(filename, "rb");

    struct stat file_stat;

    if (file == NULL || fstat(fileno(file), &file_stat) == -1 || S_ISDIR(file_stat
        .st_mode)) {
        printf(WARN "File␣" E_ITALIC "%s" E_RESET "␣could␣not␣open,␣sending␣404␣
            response\n", filename);
        const char* resp =
        "HTTP/1.1␣404␣Not␣Found\n"
        "Server:␣Prikol\n"
        "Connection:␣Close\n"
        "\n"
        "\n";
        send(socketfd, resp, sizeof(resp), 0);
        return EXIT_FAILURE;
    }


    printf(INFO "downloading␣" E_ITALIC "%s" E_RESET "␣in␣memory\n", filename);

    char* content = (char*)malloc((file_stat.st_size + 1) * sizeof(*content));
    content[file_stat.st_size] = '\0';

    size_t actual_length_readed = fread(content, 1, file_stat.st_size, file);

    if (actual_length_readed == file_stat.st_size)
        printf(INFO "content␣length:␣%ld\n", file_stat.st_size);
    else
        printf(WARN "content␣length␣is␣%ld,␣but␣read␣only␣%ld\n", file_stat.
            st_size, actual_length_readed);


    fclose(file);

    const char* content_type = define_content_type(filename);

    printf(INFO "i␣think␣" E_ITALIC TOSTRING(content_type) E_RESET "␣is␣\'%s\'\n",
         content_type);


    char* response;
```

```
    size_t response_len;

    if (1) {

        response = (char*)malloc((file_stat.st_size + 2048) * sizeof(*response));
        response_len = sprintf(response,
            "HTTP/1.1 200 OK"
            "Content-Type: %s"
            "Content-Length: %ld"
            "\n"
            "\n", content_type, file_stat.st_size
        );

        printf(INFO "response is ready\n");

        printf(E_RESET INFO "sending...\n");


        size_t ret = send(socketfd, response, response_len, MSG_MORE);
        ret += send(socketfd, content, file_stat.st_size, 0);
        return ret;
    }
    printf(INFO "response is ready\n");

    printf(E_RESET INFO "sending...\n");


    return send(socketfd, response, response_len, 0);

}









const char* define_content_type(const char* filename) {

    if (find(filename, ".html") != -1) return "text/html";
    if (find(filename, ".svg" ) != -1) return "image/svg+xml";
    if (find(filename, ".png" ) != -1) return "image/png";
    if (find(filename, ".jpg" ) != -1) return "image/jpeg";
    if (find(filename, ".js"  ) != -1) return "text/javascript";
    if (find(filename, ".css" ) != -1) return "text/css";
    if (find(filename, ".json") != -1) return "text/json";

    return "text/plain";
}
```

13 Приложение К: листинг файла ./src/server.c

```
#include "server.h"
#include "internet.h"
#include "messages.h"
#include "list.h"
#include "http.h"

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

#include "transformer.h"


long find(const char* from, const char* what) {
    for (size_t i = 0; from[i] != '\0'; ++i) {
        char correct = 1;
        for (size_t j = 0; what[j] != '\0'; ++j) {
            if (from[i + j] == '\0' || from[i + j] != what[j]) {
                correct = 0;
                break;
            }
        }
        if (correct) return i;
    }
    return -1l;
}




int serve_client(int socketfd) {

    if (-1 == socketfd) return -1;

    printf(INFO "new␣connection\n");


    http_message_additional msg = get_http_message(socketfd);

    printf(E_BOLD "received:" E_RESET "\n");
    printf(E_ITALIC);
    printf("method:␣␣␣%s\n", msg.message.method);
    printf("url:␣␣␣␣␣␣%s\n", msg.message.url);
    for (size_t i = 0; i < msg.message.url_args.length; ++i) {
        printf("\t%s=%s&\n", msg.message.url_args.data[i].key, msg.message.
            url_args.data[i].value);
    }
    printf("http␣ver:␣%s\n", msg.message.http_version);
    printf("headers:\n");
    for (size_t i = 0; i < msg.message.headers.length; ++i) {
        printf("\t%s:␣%s\n", msg.message.headers.data[i].key, msg.message.headers.
            data[i].value);
    }
    printf("body:␣%s\n\n", msg.message.body);
    printf(E_RESET "\n");
```

```
printf(INFO "i␣receive:\n");

char* url = msg.message.url;

printf("\turl:␣%s\n", url);

char* filename = strcpy_a(url, "public");

printf("\tpath:␣%s\n", filename);


enum METHOD_E method = get_method(msg.message.method);

printf("\tmethod:␣%d\n", method);


if (method == GET) {

    printf(INFO "i␣think␣client␣wants␣my␣file␣" E_ITALIC "%s" E_RESET "\n",
        filename);

    send_small_file(socketfd, filename);


} else if (method == CALC) {



    printf(INFO "client␣wants␣to␣calculate␣this␣silly␣equation␣" E_ITALIC "%s"
         E_RESET "\n", msg.message.body);

    list_transformation_t transforms = get_all_transformations_s(msg.message.
        body);

    char* response = calloc(100000, sizeof(char));
    char* content = calloc(100000, sizeof(char));

    size_t content_length = sprintf(content,
        "<?xml␣version=\"1.0\"␣encoding=\"UTF-8\"?>\n"
        "<transforamtions>\n"
            "<transformation>\n"
                "<description>%s</description>\n"
                "<solution>%s</solution>\n"
            "</transformation>\n"
        "</transforamtions>",
        transforms.data[0].comment,
        get_string_from_symbol(transforms.data[0].equation.symbols.data[0])
    );

    size_t len = sprintf(response,
        "HTTP/1.1␣200␣OK\n"
        "Content-Type:␣application/xml\n"
        "Content-Length:␣%ld\n"
        "\n"
```

```
            "%s",
            content_length,
            content
        );



        send(socketfd, response, len, 0);



        free(response);
        free(content);

    } else {

        printf(WARN "   ␣   ␣   ");
        const char* resp =
        "HTTP/1.1␣501␣Not␣Implemented\n"
        "Connection:␣Close\n"
        "\n"
        "\n";
        send(socketfd, resp, sizeof(resp), 0);

    }

    free(filename);
    free(msg.alloced_raw_content);
    close(socketfd);
    printf(INFO "session␣end\n\n");

    return 0;

}









char* read_all_alloc(size_t* len, int socketfd) {
    size_t allocated = 0;
    *len = 0;
    char* data = (char*)malloc(allocated * sizeof(*data));
    do {
        allocated += 1024;
        data = (char*)realloc(data, allocated + 1);
        ssize_t read_len = read(socketfd, data + *len, allocated - *len);
        if (-1 == read_len) return NULL;
        *len += read_len;
        if ((size_t)read_len == (allocated - *len - read_len)) break;
        if (read_len == 0) break;
        if (read_len < 0) {
            free(data);
```

```
            *len = 0;
            return NULL;
        }
    } while (0);
    data[*len] = '\0';
    return data;
}

void write_all(char* data, size_t data_len, int socketfd) {

}
```

14 Приложение Л: листинг файла ./src/server.h

```
#pragma once

#include <stdlib.h>


char* read_all_alloc(size_t* len, int socketfd);
void write_all(char* data, size_t data_len, int socketfd);


int serve_client(int socketfd);


int send_small_file(int socketfd, const char* filename);
```

15 Приложение М: листинг файла ./src/transformer.c

```


#include <stdio.h>
#include "transformer.h"


struct symbol_kv {
    const char* key;
    enum symbol_type value;
};

struct symbol_kv symbols_table[] = {

    // { "NULL"               , NULL_SYMBOL },



    { " "                  , IMPLIES },
    { "\\rightarrow"         , IMPLIES },
    { "\\Rightarrow"         , IMPLIES },
    { "\\imp"                , IMPLIES },

    { " "                  , EQUIVALENT },
```

```
    { "\\eq"                , EQUIVALENT },

    { "¬"                   , NOT },
    { "\\not"               , NOT },

    { " "                   , AND },
    { "\\and"               , AND },

    { " "                   , OR },
    { "\\or"                , OR },

    { " "                   , XOR },
    { "\\xor"               , XOR },



    { " "                   , TRUE },
    { "\\top"               , TRUE },
    { "\\true"              , TRUE },

    { " "                   , FALSE },
    { "\\bot"               , FALSE },
    { "\\false"             , FALSE },



    { " "                   , FORALL },
    { "\\forall"            , FORALL },

    { " "                   , EXIST },
    { "\\exist"             , EXIST },



    { "("                   , BRACE_OPEN_ROUND },
    { ")"                   , BRACE_CLOSE_ROUND },



    { "+"                   , PLUS },
    { "-"                   , MINUS },
    { "*"                   , MULTIPLY },
    { "/"                   , DIVIDE },

};
#define LEN(x) (sizeof(x) / sizeof(*(x)))


const char* get_string_from_symbol_type(enum symbol_type type) {
    for (size_t i = 0; i < LEN(symbols_table); ++i) {
        if (symbols_table[i].value == type) return symbols_table[i].key;
    }
    if (type == VARIABLE) return "VARIABLE";
    if (type == NUMBER) return "NUMBER";
    return "NULL_TYPE";
}
```

```
char* soviet_union = NULL;
size_t soviet_union_len = 0;

const char* get_string_from_symbol(struct symbol symbol) {
    for (size_t i = 0; i < LEN(symbols_table); ++i) {
        if (symbols_table[i].value == symbol.type) return symbols_table[i].key;
    }
    if (symbol.text != NULL) {
        if (soviet_union == NULL) {

            soviet_union_len = symbol.text_len+1;
            soviet_union = (char*)malloc(soviet_union_len);

        } else if (soviet_union_len-1 < symbol.text_len) {

            soviet_union_len = symbol.text_len+1;
            soviet_union = (char*)realloc(soviet_union, soviet_union_len);

        }
        memcpy(soviet_union, symbol.text, symbol.text_len + 1);
        soviet_union[symbol.text_len] = '\0';
        return soviet_union;
    }
    return "NULL_SYMBOL";
}


int is_word_symbol(char c) {
    return (
        (c >= 'a' && c <= 'z') ||
        (c >= 'A' && c <= 'Z') // ||
        // (c >= '0' && c <= '1') ||
        // (c == '_')
    );
}

const char*     get_next_word(const char* string, size_t* spaces, size_t* word_len
    , enum symbol_type* type) {

    for (*spaces = 0; string[(*spaces)] != '\0'; ++(*spaces))
        if (
            string[(*spaces)] !=  '␣' &&
            string[(*spaces)] != '\t' &&
            string[(*spaces)] != '\n'
        ) break;
    string = string + *spaces;



    for (*word_len = 0; is_word_symbol(string[*word_len]); ++(*word_len)) {}

    if (0 == *word_len) return NULL;
    *type = VARIABLE;
    return string;
```

```
}


const char*     get_next_operator(const char* string, size_t* spaces, size_t*
    word_len, enum symbol_type* type) {

    for (*spaces = 0; string[(*spaces)] != '\0'; ++(*spaces))
        if (
            string[(*spaces)] !=  '␣' &&
            string[(*spaces)] != '\t' &&
            string[(*spaces)] != '\n'
        ) break;
    string = string + *spaces;

    for (size_t i = 0; i < LEN(symbols_table); ++i) {
        int accept = 1;
        *word_len = 0;
        for (size_t j = 0; symbols_table[i].key[j] != '\0'; ++j) {
            if (symbols_table[i].key[j] != string[j]) {
                accept = 0;
                break;
            }
            if (symbols_table[i].key[j + 1] == '\0') *word_len = j + 1;
        }
        if (accept) {
            *type = symbols_table[i].value;
            return string;
        }
    }

    return NULL;

}



const char*     get_next_number(const char* string, size_t* spaces, size_t*
    word_len, enum symbol_type* type) {

    for (*spaces = 0; string[(*spaces)] != '\0'; ++(*spaces))
        if (
            string[(*spaces)] !=  '␣' &&
            string[(*spaces)] != '\t' &&
            string[(*spaces)] != '\n'
        ) break;
    string = string + *spaces;


    int was_dot = 0;
    for (*word_len = 0; string[*word_len] != '\0'; ++(*word_len)) {

        if (string[(*word_len)] == '.') {
            if (was_dot) break;
            else {
                was_dot = 1;
                continue;
```

```
            }
        }

        if (string[*word_len] < '0' || string[*word_len] > '9') break;


    }


    if (0 == *word_len) return NULL;
    *type = NUMBER;
    return string;

}




struct symbol   get_next_symbol(const char* string, size_t* spaces, size_t*
    text_len) {


    struct symbol symb = {
        .type = NULL_SYMBOL,
        .text = NULL,
        .text_len = 0,
    };


    const char* nw;
    enum symbol_type type;


    nw = get_next_operator(string, spaces, text_len, &type);

    if (nw != NULL) {
        symb.text = nw;
        symb.text_len = *text_len;
        symb.type = type;
        return symb;
    }



    nw = get_next_word(string, spaces, text_len, &type);

    if (nw != NULL) {
        symb.text = nw;
        symb.text_len = *text_len;
        symb.type = type;
        return symb;
    }



    nw = get_next_number(string, spaces, text_len, &type);
```

```
    if (nw != NULL) {
        symb.text = nw;
        symb.text_len = *text_len;
        symb.type = type;
        return symb;
    }


    return symb;
}


struct equation get_equation_from_string(const char* string) {

    struct equation eq = {0};

    size_t spaces = 0;
    size_t word_length = 0;

    for (size_t i = 0; string[i] != '\0'; i += (spaces + word_length)) {
        struct symbol next_symbol = get_next_symbol(string + i, &spaces, &
            word_length);

        // printf("nextw symblol: %ld (%s), spaces: %ld, word_length: %ld\n",
        //     next_symbol.type,
        //     get_string_from_symbol_type(next_symbol.type),
        //     spaces,
        //     word_length
        // );

        if (next_symbol.type) {
            LIST_APPEND(next_symbol, eq.symbols);
        }
        else spaces = 1;
    }


    return eq;

}






unsigned char get_operation_proirity(enum symbol_type symb) {
    return (symb >> (8*2)) & 0b11111111;
}

#define PRINT_SYMBOL_LIST(list) \
        for (size_t s = 0; s < list.length; ++s)\
        {\
            if (where_paste == s) printf("_␣");\
            printf("%s␣", get_string_from_symbol(list.data[s]));\
        }\
```

```
        printf("\n");


struct list_symbol_t braces_to_reverse_polish(const struct list_symbol_t symbols,
    size_t from, size_t* processed_count) {

    struct list_symbol_t symbols_r = {0};
    size_t where_paste = symbols_r.length; // which is 0
    size_t previous_braces_end = 0;

    for (size_t i = from; i < symbols.length; ++i) {
        if (symbols.data[i].type & VARIABLE) {

            // printf("RP: %3ld [[ %s ]] ", i, get_string_from_symbol(symbols.data
                [i]));
            LIST_PUSH_AT(symbols.data[i], symbols_r, where_paste);
            where_paste = symbols_r.length;
            // PRINT_SYMBOL_LIST(symbols_r);

        } else if (symbols.data[i].type & OPERATOR) {

            // printf("RP: %3ld [[ %s ]] ", i, get_string_from_symbol(symbols.data
                [i]));
            LIST_APPEND(symbols.data[i], symbols_r);
            where_paste = symbols_r.length - 1; // useless

            // printf("RP: SWAP? %s and %s\n",
            //     (symbols_r.data[symbols_r.length-2].type & OPERATOR) ? "true" :
                 "false",
            //     (get_operation_proirity(symbols_r.data[symbols_r.length-2].type
                ) < get_operation_proirity(symbols_r.data[symbols_r.length-1].type
                )) ? "true" : "false"
            // );
            // PRINT_SYMBOL_LIST(symbols_r);
            if ((symbols_r.data[symbols_r.length-2].type & OPERATOR)
                && (previous_braces_end < symbols_r.length-2)
                && (get_operation_proirity(symbols_r.data[symbols_r.length-2].type
                    ) < get_operation_proirity(symbols_r.data[symbols_r.length-1].
                    type))
            )
            {
                // printf("RP: %3ld %7s ", i, "SWAP!");
                SWAP(symbols_r.data[symbols_r.length-1], symbols_r.data[symbols_r.
                    length-2], symbol_t);
                where_paste -= 1;
                // PRINT_SYMBOL_LIST(symbols_r);
            }

        } else if (symbols.data[i].type == BRACE_OPEN_ROUND) {

            // printf("RP: %3ld [[ ( ]]\n", i);
            size_t pc = 0;
            struct list_symbol_t braces_stuff = braces_to_reverse_polish(symbols,
                i+1, &pc);
            i += pc;
```

```
            LIST_PUSH_RANGE_AT(braces_stuff.data, braces_stuff.length, symbols_r,
                where_paste);
            previous_braces_end = where_paste + braces_stuff.length - 1;
            LIST_FREE(braces_stuff);
            where_paste = symbols_r.length;
            // PRINT_SYMBOL_LIST(symbols_r);

        } else if (symbols.data[i].type == BRACE_CLOSE_ROUND) {

            // printf("RP: %3ld [[ ) ]] ", i);
            if (processed_count) *processed_count = i - from + 1;
            return symbols_r;
        }
    }
    if (processed_count) *processed_count = symbols.length - from;
    return symbols_r;

}

struct equation  to_postfix_notation(struct equation eq) {

    struct equation eq_r = {.symbols = braces_to_reverse_polish(eq.symbols, 0,
        NULL)};
    return eq_r;

}


struct equation  to_infix_notation(struct equation reverse_polish) {
    // a b + c - g *
    //
    // ( ( ( a + b ) - c ) * g )

    // a b c + *
    // ( b + c ) * c

    symbol_t brace_open = {
        .type = BRACE_OPEN_ROUND,
        .text = "(",
        .text_len = 1,
    };

    symbol_t brace_close = {
        .type = BRACE_CLOSE_ROUND,
        .text = ")",
        .text_len = 1,
    };


    struct equation norm_r = {0};

    LIST(symbol_t) symbol_stack = {0};

    for (size_t i = 0; i < reverse_polish.symbols.length; ++i) {
        if (reverse_polish.symbols.data[i].type & OPERAND) {
            LIST_APPEND(reverse_polish.symbols.data[i], symbol_stack);
```

```
        } else {
            if (norm_r.symbols.length > 0) {
                LIST_PUSH_AT(brace_open, norm_r.symbols, 0);
                LIST_APPEND(brace_close, norm_r.symbols);
            }

            size_t push_at = norm_r.symbols.length;

            if (symbol_stack.length > 0) {
                LIST_PUSH_AT(symbol_stack.data[symbol_stack.length-1], norm_r.
                    symbols, push_at);
                symbol_stack.length -= 1;
            }

            LIST_PUSH_AT(reverse_polish.symbols.data[i], norm_r.symbols, push_at);

            if (symbol_stack.length > 0) {
                LIST_PUSH_AT(symbol_stack.data[symbol_stack.length-1], norm_r.
                    symbols, push_at);
                symbol_stack.length -= 1;
            }


        }
    }

    return norm_r;
}


double do_algebra(double n1, double n2, enum symbol_type oper) {
    switch (oper) {
        case PLUS:
            return n1 + n2;
            break;
        case MINUS:
            return n1 - n2;
            break;

        case MULTIPLY:
            return n1 * n2;
            break;
        case DIVIDE:
            return n1 / n2;
            break;

        default:
            return 0;
    }
    return 0;
}

double compute(struct equation* postfix) {

    LIST(double) number_stack = {0};
```

```
    for (size_t i = 0; i < postfix->symbols.length; ++i) {
        if (postfix->symbols.data[i].type & ALGEBRAIC_OPERATOR) {
            number_stack.data[number_stack.length-2] = do_algebra(number_stack.
                data[number_stack.length-2], number_stack.data[number_stack.length
                -1], postfix->symbols.data[i].type);
            number_stack.length -= 1;
        } else {
            double number;
            char* number_str = (char*)calloc(postfix->symbols.data[i].text_len+1,
                sizeof(*number_str));
            strcpy(number_str, postfix->symbols.data[i].text);
            number = atof(number_str);
            free(number_str);
            LIST_APPEND(number, number_stack);
        }
    }

    return number_stack.data[0];

}

int computable(struct equation* postfix, double* solution) {

    for (size_t i = 0; i < postfix->symbols.length; ++i) {
        if (!(
            (postfix->symbols.data[i].type & ALGEBRAIC_OPERATOR) ||
            (postfix->symbols.data[i].type & NUMBER)
        )) return 0;
    }

    *solution = compute(postfix);
    return 1;

}




const struct equation_tree null_tree = NULL_TREE;

list_transformation_t get_all_transformations_s(const char* string) {
    struct equation eq = get_equation_from_string(string);
    list_transformation_t r = get_all_transformations(&eq);
    LIST_FREE(eq.symbols);
    return r;
}



list_transformation_t get_all_transformations(struct equation* equation) {

    list_transformation_t transformations = {0};

    struct equation postfix = to_postfix_notation(*equation);

    double solution;
```

```
    if (computable(&postfix, &solution)) {
        struct equation solution_equation = {0};
        struct symbol solution_symbol = {
            .text = (char*)malloc(20), // omg
            .text_len = 20,
            .type = NUMBER,
        };
        solution_symbol.text_len = snprintf(solution_symbol.text, 20, "%lf",
            solution);
        LIST_APPEND(solution_symbol, solution_equation.symbols);
        transformation_t solution_transformation = {
            .equation = solution_equation,
            .comment = "solve the problem",
        };
        LIST_APPEND(solution_transformation, transformations);
    }

    // struct equation_tree eq_tree = null_tree;
    // for (size_t i = equation->symbols.length-1; i < equation->symbols.length;
        --i) {
    //     add_symbol_to_equation_tree(equation->symbols.data[i], &eq_tree);
    // }


    return transformations;

}



unsigned char get_operator_arity(enum symbol_type oper) {
    if (oper & OPERATOR) {
        return ((oper >> 8) & 0b11111111);
    }
    return 0;
}



void add_symbol_to_equation_tree(symbol_t symbol, struct equation_tree* tree) {

    if (tree->symbol.type == NULL_SYMBOL) {
        tree->symbol = symbol;
        if (tree->symbol.type & VARIABLE) {
            tree->filled = 1;
            return;
        }
        LIST_INIT_LEN(tree->sub_equations, get_operator_arity(tree->symbol.type));
        goto check;
    }

    if (tree->symbol.type & VARIABLE) {
        return;
    }
```

```
    for (size_t i = tree->sub_equations.alloc_length-1; i < tree->sub_equations.
        alloc_length; --i) {
        if (!tree->sub_equations.data[i].filled) {
            add_symbol_to_equation_tree(symbol, tree->sub_equations.data + i);
            if (tree->sub_equations.data[i].filled)
                tree->sub_equations.length = tree->sub_equations.alloc_length - i;
            goto check;
        }
    }
check:

    if (tree->sub_equations.length >= get_operator_arity(tree->symbol.type))
        tree->filled = 1;


}

void print_equation_tree(struct equation_tree* tree, size_t zindex) {
    for (size_t i = 0; i < zindex; ++i) printf("␣␣␣");
    printf("[%s]", get_string_from_symbol(tree->symbol));
    if (tree->sub_equations.length != (size_t)get_operator_arity(tree->symbol.type
        ))
        printf("␣(sub␣length:␣%ld,␣but␣would␣be␣%ld)",
            tree->sub_equations.length,
            (size_t)get_operator_arity(tree->symbol.type)
        );
    putchar('\n');
    for (size_t i = 0; i < tree->sub_equations.length; ++i)
        print_equation_tree(tree->sub_equations.data + i, zindex+1);
}
```

16 Приложение H: листинг файла ./src/transformer.h

```
#pragma once

#include <stdlib.h>
#include "list.h"



enum symbol_type {

    NULL_SYMBOL                 = 0
        b00000000000000000000000000000000000000000000000000000000000000000,


    IMPLIES                     = 0
        b10000000000000000000000000000000000010110000000100000010000000001,
    EQUIVALENT                  = 0
        b10000000000000000000000000000000000010010000000100000010000000010,
    NOT                         = 0
        b10000000000000000000000000000000000000000010000010000000010000011,
    AND                         = 0
        b10000000000000000000000000000000000000010000000001000000100000100,
```

```
    OR                          = 0
        b10000000000000000000000000000000000011100000000100000010000000101,
    XOR                         = 0
        b10000000000000000000000000000000000011000000001000000010000000110,

    TRUE                        = 0
        b01000000000000000000000000000000000000000000000000000100000001,
    FALSE                       = 0
        b01000000000000000000000000000000000000000000000000000000000000,

    FORALL                      = 0
        b00100000000000000000000000000000000000000000100000000100000001,
    EXIST                       = 0
        b00100000000000000000000000000000000000000000100000000100000010,
    // NOEXIST                      = 0
        b00000011000000000000000000000000000000000000000000000000000011,

    BRACE_OPEN_ROUND            = 0
        b00010000000000010000000000000000000000000000000000000000000000,
    BRACE_CLOSE_ROUND           = 0
        b00010000000000010000000000000000000000000000000000000000000001,

    // BRACE_OPEN_FIGURE            = 0
        b00000101000000000000000000000000000000000000000000000000000000,
    // BRACE_CLOSE_FIGURE           = 0
        b00000101000000000000000000000000000000000000000000000000000001,



    OPERAND                     = 0
        b00001000000000000000000000000000000000000000000000000000000001,

    VARIABLE                    = 0
        b00001000000000000000000000000000000000000000000000000000000000,
    NUMBER                      = 0
        b00001000000000000000000000000000000000000000000000000000000001,

    OPERATOR                    = 0
        b10100100000000000000000000000000000000000000000000000000000000,

    ALGEBRAIC_OPERATOR          = 0
        b00000100000000000000000000000000000000000000000000000000000000,

    PLUS                        = 0
        b00000100000000000000000000000000000000000000100000010000000000,
    MINUS                       = 0
        b00000100000000000000000000000000000000000000100000010000000001,

    MULTIPLY                    = 0
        b00000100000000000000000000000000000000000000100000010000000010,
    DIVIDE                      = 0
        b00000100000000000000000000000000000000000000100000010000000011,

    NEGATE                      = 0
        b00000100000000000000000000000000000000000000110000000100000100,
};
```

```
struct symbol {
    enum symbol_type type;

    const char* text;
    size_t text_len;
};
typedef struct symbol symbol_t;

LIST(symbol_t);

struct equation {
    struct list_symbol_t symbols;
};

typedef struct equation equation_t;




struct equation_tree {
    symbol_t symbol;
    struct list_equation_tree_t {
        struct equation_tree* data;
        size_t length;
        size_t alloc_length;
    } sub_equations;
    int filled;
};

#define NULL_TREE {\
    .symbol = NULL_SYMBOL,\
    .sub_equations = {0},\
    .filled = 0\
}

void add_symbol_to_equation_tree(symbol_t symbol, struct equation_tree* tree);
void print_equation_tree(struct equation_tree* tree, size_t zindex);


const char* get_string_from_symbol_type(enum symbol_type type);
const char* get_string_from_symbol(struct symbol symbol);


struct symbol   get_next_symbol(const char* string, size_t* spaces, size_t*
    text_len);
struct equation get_equation_from_string(const char* string);


struct equation  to_postfix_notation(struct equation eq);
struct equation  to_infix_notation(struct equation reverse_polish);


int computable(struct equation* postfix, double* solution);
```

```
typedef struct transformation {
    equation_t equation;
    const char* comment;
} transformation_t;

typedef LIST(transformation_t) list_transformation_t;

list_transformation_t get_all_transformations_s(const char* string);
list_transformation_t get_all_transformations(struct equation* equation);
```

17 Приложение O: листинг файла ./src/transformer_test.c

```
#include <stdio.h>

#include "transformer.h"
#include "list.h"


#define ARR_LEN(x) (sizeof(x) / sizeof(*(x)))

int main() {


    // LIST(int) list = {0};

    // LIST_APPEND(10, list, int);
    // LIST_APPEND(20, list, int);
    // LIST_APPEND(30, list, int);
    // LIST_APPEND(40, list, int);
    // LIST_APPEND(50, list, int);



    // int arr[] = {1, 2, 3, 4};
    // const size_t arr_len = sizeof(arr) / sizeof(*arr);

    // LIST_PUSH_RANGE_AT(arr, arr_len, list, arr[1], int);

    // for (size_t i = 0; i < list.length; ++i) printf("%2d; ", list.data[i]);
        putchar('\n');

    // return 0;

    // printf("computing standart permutations... ");

    // struct equation permuts[] = {
    //     get_equation_from_string("a + b \\eq b + a"),
    //     get_equation_from_string("a + (b + c) \\eq (a + b) + c"),

    //     get_equation_from_string("a + 0 \\eq a"),
    //     get_equation_from_string("a + (-a) \\eq 0"),

    //     get_equation_from_string("a * b \\eq b * a"),
    //     get_equation_from_string("a * (b * c) \\eq (a * b) * c"),
```

```
//     get_equation_from_string("1 * a \\eq a"),
//     get_equation_from_string("a * (1/a) \\eq 1"),
//     get_equation_from_string("(a + b) * c \\eq a * c + b * c"),

//     // get_equation_from_string("a = b"),
//     // get_equation_from_string("a \leq c"),
//     // get_equation_from_string("a + c \leq b + c"),
//     // get_equation_from_string("0 \leq a * b"),
//     // get_equation_from_string("a < b"),
//     // get_equation_from_string("b \geq a"),
//     // get_equation_from_string("a > b"),

//     // get_equation_from_string("\exists c \in \mathbb{R} | a \leq c \land
    c \leq b"),
// };
// printf("done.\n");



char buffer[2056];

while (1) {

    printf("your equation\n> ");

    fgets(buffer, sizeof(buffer), stdin);

    printf("\n\n");

    // size_t transformations;

    // struct equation* equations = get_all_transformations_s(&transformations
        , buffer);

    struct equation eq = get_equation_from_string(buffer);

    printf("received: ");

    for (size_t i = 0; i < eq.symbols.length; ++i) {
        printf("%s ", get_string_from_symbol(eq.symbols.data[i]));
    }
    printf("\n\n");


    struct equation eq_p = to_postfix_notation(eq);

    printf("to reverse polish from normal: ");

    for (size_t i = 0; i < eq_p.symbols.length; ++i) {
        printf("%s ", get_string_from_symbol(eq_p.symbols.data[i]));
    }
    printf("\n\n");


    double result;
    if (computable(&eq_p, &result))
```

```
            printf("solution: %lf\n\n", result);



        struct equation eq_np = to_infix_notation(eq_p);

        printf("to normal from reverse polish: ");

        for (size_t i = 0; i < eq_np.symbols.length; ++i) {
            printf("%s ", get_string_from_symbol(eq_np.symbols.data[i]));
        }
        printf("\n\n");




        printf("equation tree from reverse polish: \n");

        struct equation_tree tree = NULL_TREE;
        for (size_t i = eq_p.symbols.length - 1; i < eq_p.symbols.length; --i) {
            add_symbol_to_equation_tree(eq_p.symbols.data[i], &tree);
        }

        print_equation_tree(&tree, 0);

        printf("\n\n");






        // size_t trans_len = 0;
        // struct equation* transformations = get_all_transformations(&eq, &
            trans_len, permuts, ARR_LEN(permuts));

        // printf("transformations:\n");
        // for (size_t i = 0; i < trans_len; ++i) {
        //     for (size_t j = 0; j < transformations[i].symbols.length; ++j) {
        //         printf("%s ", get_string_from_symbol(transformations[i].symbols
            .data[j]));
        //     }
        //     printf("\n");
        // }




    }

}
```

18 Приложение П: листинг файла ./public/equation-list.css

```
```

```
.equations {
    display: flex;
    flex-wrap: wrap;
}

.equation {
    margin: 10px;
    width: 100%;
    border: 1px solid #e1e1e142;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    padding: 20px;
}

.title {
    font-size: 1.6rem;
    margin-top: 10px;
    margin-bottom: 10px;
    font-weight: 900;
    text-align: center;
    border-bottom: 1px dashed #e1e1e16e;
    padding-bottom: 10px;
}

.description {
    font-size: 1.2rem;
    margin-top: 10px;
    margin-bottom: 10px;
    border-bottom: 1px dashed #e1e1e16e;
    padding-bottom: 10px;
}
```

19 Приложение P: листинг файла ./public/equations.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
    <title   >       </title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/katex@0.13.11/dist/
        katex.min.css">
    <script defer src="https://cdn.jsdelivr.net/npm/katex@0.13.11/dist/katex.min.
        js"></script>
    <script defer src="https://cdn.jsdelivr.net/npm/katex@0.13.11/dist/contrib/
        auto-render.min.js"></script>

    <link rel="shortcut␣icon" href="favicon.png" type="image/x-icon">

    <link rel="stylesheet" href="global.css">
    <link rel="stylesheet" href="menu.css">
    <link rel="stylesheet" href="equation-list.css">
</head>
<body>
```

```
    <a href="index.html"><div class="logo"><img src="logo.svg"></div></a>

    <div class="container">

        <div class="transitions-container" class="equations" id="xmlContent">

        </div>

    </div>

    <script>

        function call_katex() {
            renderMathInElement(document.body, {
                delimiters: [
                    {left: "$", right: "$", display: false},
                    {left: "\\[", right: "\\]", display: true}
                ]
            });
        };

        //      XML
        fetch('equations.xml')
            .then(response => response.text())
            .then(data => {
                //          XML
                const parser = new DOMParser();
                const xmlDoc = parser.parseFromString(data, 'application/xml');

                //             XML
                const equations = xmlDoc.getElementsByTagName('equation');
                let output = '<div␣class="equations">';

                for (let i = 0; i < equations.length; i++) {
                    const title = equations[i].getElementsByTagName('title')[0].
                        textContent;
                    const description = equations[i].getElementsByTagName('
                        description')[0].textContent;
                    const proof = equations[i].getElementsByTagName('proof')[0].
                        textContent;

                    output += `<div class="equation"><div class="title">${title}</
                        div><div class="description">${description}</div><div
                        class="proof">${proof}</div></div>`;
                }

                output += '</div>';

                //
                document.getElementById('xmlContent').innerHTML = output;

                call_katex();
            })
            .catch(error => console.error('Error:', error));
    </script>
</body>
```

```
</html>
```

20 Приложение C: листинг файла ./public/equations.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<equations>
  <equation>
    <title   >     </title>
    <description>
              (                 :D )
    </description>
    <proof>

      $a$   $b$          $c$.
            ,
          $a$                $b$,
             $c^2$.


       ,
           ,
       :

      \[ a^2 + b^2 = c^2 \]

       ,                .
    </proof>
  </equation>

  <equation>
    <title   >     </title>
    <description>
      $(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$,    $\binom{n}{k}$
                            "$n$␣ ␣$k$",              : $\binom{n}{k} = \frac{n!}{k
          !(n-k)!}$
    </description>
    <proof>
        :

       :    $n = 1$                 :
      \[
      (a + b)^1 = a^1 + \binom{1}{1}a^{1-1}b^1 = a + b
      \]
       :                $n = k$,  ..
      \[
      (a + b)^k = \sum_{i=0}^{k} \binom{k}{i} a^{k-i}b^i
      \]
       :     ,                $n = k + 1$:

      $ (a + b)^{k+1} = (a + b)(a + b)^k $
      $ = (a + b) \sum_{i=0}^{k} \binom{k}{i} a^{k-i}b^i $
      $ = \sum_{i=0}^{k} \binom{k}{i} a^{k+1-i}b^i + \sum_{i=0}^{k} \binom{k}{i} a
          ^{k-i}b^{i+1} $
      $ = \binom{k}{0}a^{k+1}b^0 + \sum_{i=1}^{k-1} \binom{k}{i} a^{k+1-i}b^i + \
          binom{k}{k}a^0b^{k+1} $
```

```
      $ = a^{k+1} + \sum_{i=1}^{k-1} \left( \binom{k}{i} + \binom{k}{i-1} \right)
          a^{k+1-i}b^i + b^{k+1} $
      $ = a^{k+1} + \sum_{i=1}^{k-1} \binom{k+1}{i} a^{k+1-i}b^i + b^{k+1} $
      $ = \sum_{i=0}^{k+1} \binom{k+1}{i} a^{k+1-i}b^i $

       ,                                    $n$.
    </proof>
  </equation>

  <equation>
    <title>

    </title>
    <description>
          $4n + 1$                              .
    </description>
    <proof>
      ,    $p$ -           $4n + 1$.                 $a$   $b$,       $p = a^2 + b
          ^2$.

        $a^2 + b^2$       4.                        4           0    1,   $a^2 + b^2
            \equiv 0, 1 \pmod{4}$.

       $a$   $b$       ,   $a^2 + b^2 \equiv 0 \pmod{4}$,            ,    $p$
          $4n + 1$.

      ,         $a$    $b$            .             ,     $a$ -         .

       $a^2 \equiv 1 \pmod{4}$,   $b^2 \equiv 0 \pmod{4}$.        , $a^2 + b^2 \
          equiv 1 \pmod{4}$.

       ,       ,                 $4n + 1$                              .
    </proof>
  </equation>

  <equation>
    <title>

    </title>
    <description>
        $f(x)$             $[a, b]$,            $c \in (a, b)$,
                                                    : $f'(c)␣=␣\frac{f(b)␣-␣f(a
            )}{b␣-␣a}$.
␣␣␣␣</description>
␣␣␣␣<proof>
␣␣␣␣␣␣    ␣   ␣$f(x)$,␣     ␣ ␣   ␣$[a,␣b]$␣ ␣       ␣ ␣    ␣$(a,␣b)$.␣ ␣   ␣
       ,␣     ␣  ␣$c␣\in␣(a,␣b)$,␣  ␣ ␣$f'(c) = \frac{f(b) - f(a)}{b - a}$.

       ,          $c$,                                                    $[a, b]$.
    </proof>
  </equation>

  <equation>
    <title>

    </title>
```

```
    <description>
          .
    </description>
    <proof>
      ,                              $p_1, p_2, \ldots, p_n$.          $N = p_1 \cdot
          p_2 \cdot \ldots \cdot p_n + 1$.

       $N$                          $p_1, p_2, \ldots, p_n$,                $N$
                                1.        ,    $N$                 ,
                               ,                $p_1, p_2, \ldots, p_n$.

       ,                                              .       ,
                                    .
    </proof>
  </equation>

  <equation>
    <title>

    </title>
    <description>
                .
    </description>
    <proof>

       $n$ -                  .                 $a$, $b$, $c$, $d$,       $n = a^2 +
            b^2 + c^2 + d^2$.

         .        :    $n = 1$,             $1 = 1^2 + 0^2 + 0^2 + 0^2$.

      ,                          $n$.          $n+1$.    $n+1$ -          ,
                     .    , $n+1$               $n+1 = a^2 + b^2 + c^2 + d^2$.

       ,                           .

    </proof>
  </equation>
<!--
  <equation>
    <title>
    </title>
    <description>
    </description>
    <proof>
    </proof>
  </equation> -->


</equations>
```

21 Приложение T: листинг файла ./public/getstarted.css

```
:root {
```

```
  --m: 4rem;

  --red: #FF6565;
  --pink: #FF64F9;
  --purple: #6B5FFF;
  --blue: #4D8AFF;
  --green: #5BFF89;
  --yellow: #FFEE55;
  --orange: #FF6D1B;

}



body {
  margin: 0;
}

.container {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  height: 100vh;
  align-items: center;

}

.presentation {
  /* background-color: var(--red); */
  display: flex;
  align-items: center;

  /* width: 300px; */
  justify-content:space-around;

  padding: 20px;
}

.toggle-switch {
  margin-right: 20px;
}

.fancy_button {
  border: calc(0.08 * var(--m)) solid transparent;
  position: relative;
  width: fit-content;
  color: #F3F3F3;
  font-size: var(--m);
  border-radius: calc(0.7 * var(--m));
  padding: calc(0.5 * var(--m)) calc(1 * var(--m));
  margin-bottom: 10vh;
  cursor: pointer;

  background-color: #121213;
  background:
```

```
    linear-gradient(#121213, #121213),
    /* linear-gradient(#121213 50%, rgba(18, 18, 19, 0.6) 80%, rgba(18, 18, 19, 0)
        ),   */
    linear-gradient(90deg, var(--orange), var(--yellow), var(--green), var(--blue)
        , var(--purple), var(--pink), var(--red));
  background-origin: border-box;
  background-clip: padding-box, border-box, border-box;
  background-size: 200%;
  animation: animate 2s infinite linear;
}

.fancy_button::before {
  content: '';
  background: linear-gradient(90deg, var(--orange), var(--yellow), var(--green),
      var(--blue), var(--purple), var(--pink), var(--red));
  height: 100%;
  width: 100%;
  border-radius: calc(0.7 * var(--m));
  position: absolute;
  top: 0%;
  left: 0%;
  z-index: -5;
  background-size: 200%;
  animation: animate 2s infinite linear;
  filter: blur(calc(0.2 * var(--m)));

  transition: filter 2s ease-out;
}

.fancy_button:hover::before {
  filter: blur(calc(1.5 * var(--m)));
}

@keyframes animate {
  0% {background-position: 0%}
  100% {background-position: -200%}
}

@media screen and (max-width: 1000px) {
  :root {
    --m: 2rem;
  }
}
```

22 Приложение У: листинг файла ./public/getstarted.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,␣initial-scale=1.0">

    <link rel="shortcut␣icon" href="favicon.png" type="image/x-icon">
```

```
    <title>Let's␣Get␣This␣Party␣Started!</title>
␣␣␣␣<link␣rel="stylesheet"␣href="getstarted.css">
␣␣␣␣<link␣rel="stylesheet"␣href="global.css">


␣␣␣␣<link␣rel="stylesheet"␣href="toggle-switch.css">
␣␣␣␣<script␣src="toggle-switch.js"></script>

</head>
<body>

␣␣␣␣<div␣class="container">

␣␣␣␣␣␣␣␣<a␣href="index.html"><div␣class="logo"><img␣src="logo.svg"></div></a>


␣␣␣␣␣␣␣␣<button␣class="fancy_button"␣onclick="window.location.href='equations.html
    ?tutorial=true';">Dive␣In!</button>


␣␣␣␣␣␣␣␣<div␣class="presentation">
␣␣␣␣␣␣␣␣␣␣␣␣<div␣class="toggle-switch␣active"␣ontoggle="window.location.href='
    index.html';"></div>
␣␣␣␣␣␣␣␣␣␣␣␣presentation␣mode
␣␣␣␣␣␣␣␣</div>

␣␣␣␣</div>


</body>
</html>
```

23 Приложение Ф: листинг файла ./public/global.css

```
* {
  margin: 0;
  padding: 0;
}


body {
  background-color: #141516;
  color: white;
}



.logo {
  text-align: center;
}
```

```
.logo>img {
  width: min(90vw, 300px);
}
```

24 Приложение Х: листинг файла ./public/global.js

25 Приложение Ц: листинг файла ./public/index.css

```
.transformations {
    display: flex;
    flex-wrap: wrap;
    flex-direction: row;
}

.transformation {
    margin: 10px;
    width: fit-content;
    border: 1px solid #e1e1e142;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    padding: 20px;
}

.description {
    font-size: 1.6rem;
    margin-top: 10px;
    margin-bottom: 10px;
    font-weight: 900;
    text-align: center;
    border-bottom: 1px dashed #e1e1e16e;
    padding-bottom: 10px;
}

.solution {
    font-size: 1.2rem;
    margin-top: 10px;
    margin-bottom: 10px;
    border-bottom: 1px dashed #e1e1e16e;
    padding-bottom: 10px;
}
```

26 Приложение Ш: листинг файла ./public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="shortcut icon" href="favicon.png" type="image/x-icon">

    <title>Mather</title>

    <script src="trabsitions.js"></script>
    <link rel="stylesheet" href="menu.css">
    <link rel="stylesheet" href="global.css">
    <link rel="stylesheet" href="index.css">

</head>
<body>

    <a href="getstarted.html"><div class="logo"><img src="logo.svg"></div></a>

    <div class="container">
      <div class="input-container">


        <textarea name="equation" id="equation" placeholder="put your equation
            here">2 + 2 * 2</textarea>
        <button onclick="produceTransitions()">produce</button>


        <script src="transitions.js"></script>

      </div>



      <div class="transitions-container" id="xmlContent"></div>


    </div>

</body>
</html>
```

27 Приложение Щ: листинг файла ./public/menu.css

```
.container {
    display: flex;
    flex-direction: column;
    justify-content:start;
    align-items: center;

    margin: 20px;
    padding: 20px;
    background-color: #ffffff1e;
    border-radius: 5px;
    box-shadow: 0 0 50px rgba(100, 100, 100, 0.2);
```

```
    min-height: 50vh;
}

.input-container {
    display: flex;

    margin-bottom: 20px;
    width: fit-content;
    height: min-content;
}

.input-container * {
    width: max-content;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.input-container > textarea {
    width: 100%;
}


.transition {
    margin-bottom: 20px;
    padding: 10px;
    background-color: #f9f9f9;
    border: 1px solid #ccc;
    border-radius: 5px;
}
```

28 Приложение Э: листинг файла ./public/toggle-switch.css

```
.toggle-switch {
    display: inline-block;

    width: 60px;
    height: 30px;
    background-color: #ccc;
    border-radius: 15px;
    position: relative;
    cursor: pointer;

    transition: background-color 0.25s ease-in-out;
}


.toggle-switch::after {
    content: '';
    width: 20px;
    height: 20px;
```

```
    background-color: #fff;
    border-radius: 50%;
    position: absolute;
    top: 50%;
    transform: translate(0, -50%);
    transition: left 0.3s;
    left: 5px;
}

.toggle-switch.active {
    background-color: #3868a0;
}

.toggle-switch.active::after {
    left: calc(100% - 25px);
}
```

29 Приложение Ю: листинг файла ./public/toggle-switch.js

```
document.addEventListener("DOMContentLoaded", () => {
    const toggle_switches = document.querySelectorAll(".toggle-switch");

    toggle_switches.forEach(item => {

        item.addEventListener("click", (e) => {

            item.dispatchEvent(new CustomEvent("toggle", {
                detail: {
                    active: item.classList.toggle("active"),
                },
            }));

        });
    });
});
```

30 Приложение Я: листинг файла ./public/transitions.js

```
const produceTransitions = () =>
{

  console.log("hello");

  let equation = document.getElementById("equation").value;

  console.log(equation);
```

```
  fetch("", {
    method: "CALC",
    body: equation
  })
  .then(response => response.text())
  .then(data => {
      //          XML
      const parser = new DOMParser();
      const xmlDoc = parser.parseFromString(data, 'application/xml');

      //             XML
      const transformations = xmlDoc.getElementsByTagName('transformation');
      let output = '<div class="transformations">';

      for (let i = 0; i < transformations.length; i++) {
          const description = transformations[i].getElementsByTagName('description
              ')[0].textContent;
          const solution = transformations[i].getElementsByTagName('solution')[0].
              textContent;

          output += `<div class="transformation"><div class="description">${
              description}</div><div class="solution">${solution}</div></div>`;
      }

      output += '</div>';

      //
      document.getElementById('xmlContent').innerHTML = output;

  })
  .catch(error => console.error('Error:', error));


}
```