Министерство науки и высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительной техники»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Технологии программирования»
на тему: «Конкурентное программирование».

Выполнил:
Студенты группы 23ВВИ2
Стрельцов А. П.

Принял:
Сорокин А.А.

13.12.2025
отлично

Пенза 2025

Содержание

Введение 3
1 Постановка задачи 4
2 Описание алгоритма программы 5
2.1 Общая схема работы 5
2.2 Разбор функций парсинга 5
2.3 Реализация асинхронной модели обработки 6
2.4 Псевдокод 6
3 Тестирование 9
Заключение 11
Список Литературы 12
Приложение 1. Визуализация HTML объекта 13
Приложение 2. Листинг 14

Введение

Современное программное обеспечение всё чаще сталкивается с задачами, требующими одновременной обработки множества операций: загрузка данных из сети, парсинг больших файлов, взаимодействие с пользовательским интерфейсом без «зависаний», работа с внешними API и т.д. В таких условиях традиционная синхронная модель выполнения — последовательная, блокирующая — оказывается неэффективной и приводит к снижению отзывчивости приложений.

Целью данной курсовой работы является практическое освоение и сравнительный анализ моделей многопоточного и асинхронного выполнения на примере разработки приложения для анализа структуры HTML-документов. В ходе работы была совершена асинхронная реализация на основе std::async и std::future, демонстрирующая более высокоуровневый и безопасный способ организации конкурентных вычислений.

В качестве предметной области выбран парсинг HTML (выделение заголовков, абзацев, элементов списков), так как он:

- представляет собой типичную CPU-нагруженную задачу, поддающуюся декомпозиции;
- наглядно демонстрирует выигрыш от параллелизма при обработке независимых фрагментов данных;
- позволяет оценить влияние синхронизации на корректность и производительность.

Реализация выполнена на языке C++ (стандарт C++17) в среде Microsoft Visual Studio 2022. Использование стандартной библиотеки (<thread>, <future>, <mutex>, <regex>) подчёркивает актуальность и зрелость встроенных средств конкурентного программирования в современном C++.

1 Постановка задачи

В рамках курса «Технология программирования» требуется разработать консольное приложение на языке C++, реализующее анализ структуры HTML-документа с использованием **асинхронного программирования**.

Основная задача — извлечение из HTML-файла следующих элементов:

- заголовков (<h1>–<h6>),
- абзацев (<p>),
- элементов маркированных списков (<li> внутри <ul>).

Особое внимание уделяется **организации конкурентного выполнения**:
вместо низкоуровневого управления потоками (std::thread) и ручной синхронизации, в работе применяется **асинхронная модель на основе std::async и std::future**, позволяющая:

- делегировать параллельное выполнение стандартной библиотеке;
- избежать рисков, связанных с неправильным вызовом join()/detach();
- обеспечить автоматическую передачу исключений в вызывающий поток;
- упростить код за счёт отказа от явного управления потоками.

2 Описание алгоритма программы

2.1 Общая схема работы

Программа запрашивает у пользователя путь к HTML-файлу и открывает его в бинарном режиме, чтобы корректно прочитать содержимое в кодировке UTF-8. Загруженные данные сохраняются в строку, из которой при необходимости удаляется сигнатура BOM. Для анализа структуры документа одновременно запускаются три асинхронные задачи — извлечение заголовков, абзацев и элементов маркированных списков. При добавлении результатов в общий контейнер обеспечивается потокобезопасность с помощью мьютекса. После завершения всех задач собранные данные выводятся в консоль, предварительно настроенную на работу с UTF-8 для корректного отображения текста.

2.2 Разбор функций парсинга

Функции парсинга реализуют извлечение структурных элементов HTML-документа с помощью регулярных выражений. Каждая функция отвечает за один тип элементов: parseHeaders — за заголовки уровней <h1>–<h6>, parseParagraphs — за абзацы <p>, parseUnorderedLists — за элементы маркированных списков <li> внутри <ul>. Для поиска совпадений используются итераторы std::sregex_iterator, что позволяет обрабатывать все вхождения шаблона в строке без явных циклов по индексам. При добавлении найденных данных в глобальный список результатов используется std::lock_guard<std::mutex> — это гарантирует исключение гонок данных при параллельной записи, обеспечивая корректность сбора информации независимо от порядка завершения задач.

2.3 Реализация асинхронной модели обработки

Параметр std::launch::async гарантирует немедленный запуск каждой задачи в отдельном потоке выполнения, что обеспечивает реальный параллелизм на многоядерных системах. Передача данных осуществляется через std::cref(content) — это предотвращает копирование строки и повышает эффективность.

Результаты работы функций собираются в глобальный контейнер results, доступ к которому защищён с помощью std::mutex, что гарантирует потокобезопасность при одновременной записи. Ожидание завершения всех задач выполняется вызовом get() для каждого std::future, при этом исключения, возникшие в задачах, корректно пробрасываются в основной поток.

Такой подход позволяет избежать ручного управления жизненным циклом потоков (std::thread::join()), снижает вероятность ошибок и делает код более выразительным и сопровождаемым — что соответствует современным принципам технологии программирования

2.4 Псевдокод

НАЧАЛО ПРОГРАММЫ

// Глобальные данные
СОЗДАТЬ список results ДЛЯ ХРАНЕНИЯ РЕЗУЛЬТАТОВ
СОЗДАТЬ мьютекс results_mutex ДЛЯ СИНХРОНИЗАЦИИ

ФУНКЦИЯ parseHeaders(строка content):
ДЛЯ КАЖДОГО совпадения с регулярным выражением для заголовков В content:
ЗАБЛОКИРОВАТЬ мьютекс results_mutex
ДОБАВИТЬ найденный заголовок В results

РАЗБЛОКИРОВАТЬ мьютекс

ФУНКЦИЯ parseParagraphs(строка content):

ДЛЯ КАЖДОГО совпадения с регулярным выражением для параграфов В content:

ЗАБЛОКИРОВАТЬ мьютекс results_mutex

ДОБАВИТЬ найденный параграф В results

РАЗБЛОКИРОВАТЬ мьютекс

ФУНКЦИЯ parseUnorderedLists(строка content):

ДЛЯ КАЖДОГО совпадения с регулярным выражением для неупорядоченных списков В content:

ДЛЯ КАЖДОГО элемента списка В текущем списке:

ЗАБЛОКИРОВАТЬ мьютекс results_mutex

ДОБАВИТЬ элемент списка В results

РАЗБЛОКИРОВАТЬ мьютекс

ГЛАВНАЯ ФУНКЦИЯ main():

УСТАНОВИТЬ кодировку консоли В UTF-8

ЗАПРОСИТЬ имя файла у пользователя

ОТКРЫТЬ файл

ЕСЛИ файл НЕ ОТКРЫТ:

ВЫВЕСТИ сообщение об ошибке

ЗАВЕРШИТЬ программу

ПРОЧИТАТЬ содержимое файла В content

ЗАКРЫТЬ файл

// Запуск параллельных задач

СОЗДАТЬ асинхронную задачу ДЛЯ parseHeaders(content)
СОЗДАТЬ асинхронную задачу ДЛЯ parseParagraphs(content)
СОЗДАТЬ асинхронную задачу ДЛЯ parseUnorderedLists(content)

ДОЖДАТЬСЯ завершения ВСЕХ асинхронных задач

// Вывод результатов
ДЛЯ КАЖДОГО элемента В results:
    ВЫВЕСТИ тип элемента И его содержимое

ЗАВЕРШИТЬ ПРОГРАММУ

3 Тестирование

Для проверки работоспособности программы был проведён функциональный тест на реальном HTML-файле 3.html, содержащем структурированный текст с заголовками разных уровней (<h1>–<h6>), абзацами (<p>) и маркированными списками (
). Программа успешно обработала файл:

- извлекла все заголовки (например, «Расписание студента», «3 семестр»);
- выделила все абзацы, включая многострочные;
- распознала элементы списков («ЭС», «Теория вероятности», «Философия» и др.).

Проверка показала, что асинхронная модель (std::async) корректно обрабатывает данные без потери информации или сбоев — результаты всех трёх задач были собраны в один список и выведены в консоль в порядке их появления в документе. Кодировка UTF-8 (включая русскоязычные символы) отображалась корректно благодаря настройке SetConsoleOutputCP(CP_UTF8). Завершение программы с кодом 0 подтверждает отсутствие ошибок выполнения.

```
Enter file name: 3.html
header       Расписание студента
header       3 семестр
paragraph       Добро пожаловать в персональное расписание студента направления «Информатика и вычислительная техника».
Здесь представлены дисциплины по семестрам обучения.
list item       ЭЭС
header       4 семестр
list item       Теория вероятности
list item       Философия
list item       Ин яз
paragraph       Осенний семестр 2-го курса. Акцент на фундаментальные дисциплины: от математики до программирования.
header       5 семестр
list item       Физическая культура
list item       Теория автоматов
paragraph       Весенний семестр 2-го курса. Включает гуманитарные и общеобразовательные предметы для всестороннего разв
ития.
list item       ДЯП
list item       ЗД
list item       ЭВМ и ПУ
list item       ООА
list item       МРОЗ
list item       ТП (Технологии программирования)
list item       ОИС
paragraph       Осенний семестр 3-го курса. Начинается углублённая подготовка по профильным дисциплинам, связанным с выч
ислительной техникой и программным обеспечением.
paragraph       Все дисциплины обязательны для зачёта или экзамена в конце семестра.
paragraph       Обратите внимание: расписание пар (время, день недели, аудитория) уточняйте в личном кабинете студента и
ли у старосты группы.
paragraph       Успешной учёбы и продуктивного семестра!
```

Рисунок 1 – результат работы программы.

Заключение

В ходе выполнения курсовой работы были закреплены и углублены навыки современной технологии программирования, в частности — проектирования и реализации конкурентных приложений на языке C++. Особое внимание было уделено таким ключевым аспектам, как организация асинхронного выполнения задач, управление разделяемыми ресурсами и обеспечение потокобезопасности при параллельной обработке данных.

Был получен практический опыт применения средств стандартной библиотеки C++ для решения реальной задачи — анализа структуры HTML-документов. В ходе разработки программы были реализованы алгоритмы извлечения структурных элементов (заголовков, абзацев и элементов списков) с использованием регулярных выражений, а также продемонстрирован эффективный подход к распараллеливанию независимых операций с помощью std::async и std::future. Это позволило избежать низкоуровневого управления потоками, снизить риск ошибок синхронизации и повысить надёжность кода.

Применение std::mutex при доступе к общему контейнеру результатов подтвердило понимание принципов защиты критических секций и предотвращения гонок данных.

Кроме того, реализация задачи способствовала развитию навыков структурированного подхода к разработке: выделение отдельных функций парсинга, чёткое разделение этапов обработки (чтение → анализ → вывод), а также использование константных ссылок для эффективной передачи данных. Разработка данного приложения стала важным шагом в освоении современных практик написания надёжного, эффективного и поддерживаемого программного обеспечения, заложив основу для дальнейшего изучения более сложных архитектур — включая работу с асинхронными I/O, корутинами (C++20) и многопоточными системами реального времени.

Список Литературы

1. Абрамян, М. Э. Программирование на языке C++: практикум / М.Э. Абрамян. - Ростов-на-Дону: Издательство ЮФУ, 2018. - 300 с

2. Буй, Д. Б. Многопоточное программирование на C++ / Д.Б. Буй. - Москва: ДМК Пресс, 2019. - 704 с.

3. Карпов, В. Е. Архитектура многопоточных приложений / В.Е. Карпов. - Москва: ДМК Пресс, 2017. - 276 с.

4. Лафоре, Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. - Санкт-Петербург: Питер, 2019. - 928 с.

5. Липпман, С. Б. C++. Для начинающих / С.Б. Липпман, Ж. Лажойе, Б.Э. Му. - Москва: Вильямс, 2019. - 1104 с.

Приложение 1. Визуализация HTML объекта

Расписание студента

Добро пожаловать в персональное расписание студента направления «Информатика и вычислительная техника». Здесь представлены дисциплины по семестрам обучения.

3 семестр

Осенний семестр 2-го курса. Акцент на фундаментальные дисциплины: от математики до программирования.

- ЭЭС
- Теория вероятности
- Философия
- Ин яз

4 семестр

Весенний семестр 2-го курса. Включает гуманитарные и общеобразовательные предметы для всестороннего развития.

- Физическая культура
- Теория автоматов
- ДЯП
- 3Д

5 семестр

Осенний семестр 3-го курса. Начинается углублённая подготовка по профильным дисциплинам, связанным с вычислительной техникой и программным обеспечением.

- ЭВМ и ПУ
- ООА
- МРОЗ
- ТП (Технологии программирования)
- ОИС

Все дисциплины обязательны для зачёта или экзамена в конце семестра.

Обратите внимание: расписание пар (время, день недели, аудитория) уточняйте в личном кабинете студента или у старосты группы.

Успешной учёбы и продуктивного семестра!

Рисунок 2– визуализация сайта

Приложение 2. Листинг

```
#include <iostream>
#include <fstream>
#include <string>
#include <regex>
#include <list>
#include <thread>
#include <mutex>
#include <Windows.h>
#include <future>

// Глобальный контейнер результатов и мьютекс
std::list<std::pair<std::string, std::string>> results;
std::mutex results_mutex;

// Функции парсинга
void parseHeaders(const std::string& content) {
    std::regex header_regex("<h([1-6])>([^<]*)</h\\1>",
std::regex_constants::icase);
    for (std::sregex_iterator i(content.begin(), content.end(),
header_regex);
        i != std::sregex_iterator(); ++i) {
        std::lock_guard<std::mutex> lock(results_mutex);
        results.push_back({ "header", i->str(2) });
    }
}

void parseParagraphs(const std::string& content) {
    std::regex paragraph_regex("<p>([^<]*)</p>",
std::regex_constants::icase);
    for (std::sregex_iterator i(content.begin(), content.end(),
paragraph_regex);
        i != std::sregex_iterator(); ++i) {
        std::lock_guard<std::mutex> lock(results_mutex);
        results.push_back({ "paragraph", i->str(1) });
    }
}

void parseUnorderedLists(const std::string& content) {
    std::regex ul_regex("<ul>(.*?)</ul>", std::regex_constants::icase |
std::regex_constants::ECMAScript);
    for (std::sregex_iterator i(content.begin(), content.end(), ul_regex);
        i != std::sregex_iterator(); ++i) {
        std::string ul_content = i->str(1);
        std::regex li_regex("<li>([^<]*)</li>",
std::regex_constants::icase);
        for (std::sregex_iterator j(ul_content.begin(), ul_content.end(),
li_regex);
            j != std::sregex_iterator(); ++j) {
            std::lock_guard<std::mutex> lock(results_mutex);
            results.push_back({ "list item", j->str(1) });
        }
    }
}

int main() {
    // 🔧 Переключаем консоль в UTF-8
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

    std::string filename;
    std::cout << "Enter file name: ";
```

```
    std::getline(std::cin, filename);

    std::ifstream file(filename, std::ios::binary);
    if (!file.is_open()) {
        std::cerr << "Error: unable to open file '" << filename << "'\n";
        return 1;
    }

    std::string content((std::istreambuf_iterator<char>(file)),
        std::istreambuf_iterator<char>());
    file.close();

    // Запускаем асинхронные задачи
    auto fut1 = std::async(std::launch::async, parseHeaders,
std::cref(content));
    auto fut2 = std::async(std::launch::async, parseParagraphs,
std::cref(content));
    auto fut3 = std::async(std::launch::async, parseUnorderedLists,
std::cref(content));

    // Ожидаем завершения
    fut1.get();
    fut2.get();
    fut3.get();

    // Вывод результатов
    for (const auto& item : results) {
        std::cout << item.first << "       " << item.second << '\n';
    }

    return 0;
}
```