Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  
«Национальный исследовательский университет   
Высшая школа экономики»

*Факультет социально-экономических и компьютерных наук*

Мусихин Данил Михайлович

**Самостоятельная работа**

*Лабораторная работа №7*

студента образовательной программы «Разработка информационных систем для бизнеса» по направлению подготовки *09.03.04 Программная инженерия*

Руководитель, к.т.н., Доцент кафедры ИТБ.

Л.Н. Лядова

Пермь, 2023 год

**Оглавление**

[1 Постановка задачи 4](#_Toc152166223)

[2 Анализ 5](#_Toc152166224)

[3 Код 9](#_Toc152166241)

# Постановка задачи

При выполнении самостоятельной работы (индивидуального задания) студент должен самостоятельно разработать приложение (используется язык C++), реализация которого должна продемонстрировать владение навыками решения задач с использованием функций операционной системы с учётом особенностей архитектуры компьютера.

В ходе выполнения индивидуального задания студент должен показать, что он

-     Может выполнить анализ задачи, определить требования к решению, отражающие специфику предметной области, условия решения.

-     Умеет использовать возможности среды программирования для разработки приложений (кодирования, отладки) и исследования свойств разрабатываемых программ.

-     Демонстрирует знание и умение выбирать типы данных и разрабатывать структуры данных для решения задач, выбирать управляющие конструкции языка программирования.

-     Умеет работать с динамическими структурами данных.

-     Владеет навыками работы с файлами.

-     Умеет использовать средства обработки исключений различных типов, обрабатывать завершение (например, при использовании разделяемых ресурсов).

-     Способен разрабатывать многопоточные приложения, использовать средства синхронизации и взаимодействия процессов и потоков.

-     Способен разрабатывать и использовать при создании приложений библиотеки.

*Базовая оценка* **8 баллов** выставляется за выполнение задания, в котором продемонстрированы все перечисленные знания, умения и навыки.

Для повышения балла – получения оценки **9–10 баллов** необходимо разработать

-     программную документацию (техническое задание и руководство пользователя),

-     программу установки, конфигурирования приложения.

*Тема задания* (предметная область проекта, в которой решается задача разработки приложения) выбирается студентом самостоятельно, по согласованию с преподавателем. Тема может быть связана с выполнением заданий по другим дисциплинам учебного плана, результаты их выполнения могут быть использованы как задел для выполнения проекта.

Отчёт о выполнении итогового проекта (самостоятельного задания) загружается в формате ZIP-файла, в который должна быть помещена папка, поименованная с использованием шифра группы и ФИО автора (например: РИС-22-1-Иванов\_ИИ), содержащая тестовый отчёт с полным описанием проекта, каждого этапа выполнения (см. критерии оценки), решения каждой задачи по всем пунктам требований:

анализ требований и постановка задачи;

проектирование структур данных и организации файлов, ввода-вывода и взаимодействия программных модулей при решении задач;

анализ возможных ошибок и их обработка;

реализация мультизадачности, управление процессами и потоками, организация их взаимодействия, синхронизация выполнения; разработка;

использование библиотек.

В отчёт включаются самодокументированные исходные коды (все функции – их назначение, параметры и результаты, данные и алгоритмы выполнения – должны быть прокомментированы) приложений и библиотек. В ту же папку должны быть помещены все папки и файлы разработанного приложения, библиотек и пр. файлы, необходимые для тестирования приложения. При разработке приложений, работающих в сети, все сценарии их использования должны быть проиллюстрированы в отчёте текстами исходных кодов и скриншотами выполнения функций. Для уменьшения объёма отчёта все рисунки в отчёте необходимо сжать.

# Анализ и проектирование

Программа представляет собой инструмент для параллельной загрузки веб-страниц по заданным URL-адресам, сохранения их содержимого в файлы и обеспечивает обработку ошибок в многопоточной среде. Ниже представлены основные аспекты программы:

## Структуры данных:

std::vector<std::string> urls: Вектор строк для хранения URL-адресов, которые будут загружены параллельно в нескольких потоках.

std::vector<std::thread> threads: Вектор объектов потоков для параллельной обработки URL-адресов.

std::string buffer: Строка для временного хранения загруженного содержимого веб-страницы перед записью в файл.

## Организация файлов:

Для каждого URL-адреса создается свой файл с уникальным именем. Имена файлов формируются на основе индекса URL в векторе. Например, "output0.html", "output1.html" и так далее.

## Организация ввода-вывода:

Ввод URL-адресов производится с клавиатуры. Пользователь вводит URL-адреса, завершая ввод строкой "0".

Загруженное содержимое веб-страницы сохраняется в соответствующий файл на диске. В случае ошибок при записи в файл программа сообщает об ошибке.

## Мультизадачность и синхронизация потоков:

Многопоточность реализована с использованием стандартной библиотеки C++ <thread>. Каждый URL-адрес обрабатывается в отдельном потоке с помощью функции downloadWebPage.

Для избежания гонок данных при записи в файлы не используются общие ресурсы между потоками, так как каждый поток сохраняет результат в свой уникальный файл.

## Использование библиотек:

Программа использует библиотеку libcurl для выполнения HTTP-запросов и загрузки веб-страниц. Интеграция с libcurl осуществляется с использованием <curl/curl.h>.

Для работы с многопоточностью используется стандартная библиотека C++ <thread>.

## Обработка ошибок:

Для обеспечения более надежной обработки ошибок в программе, был внесен блок try-catch в функцию downloadWebPage. Этот блок перехватывает исключения типа std::system\_error и std::exception, а также любые другие неотловленные исключения (перехват через catch (...)).

В случае возникновения ошибок в потоке, программа выводит соответствующее сообщение об ошибке в стандартный поток ошибок (std::cerr), предоставляя информацию о характере ошибки и контексте, в котором она произошла. Это повышает отказоустойчивость программы и помогает в выявлении и устранении проблем в случае возникновения ошибок при загрузке и обработке веб-страниц.

Использование std::system\_error и библиотеки <system\_error> обеспечивает подробную информацию об ошибках, включая код ошибки и описание, что улучшает диагностику и отладку.

## Реализация:

### Работа программы

Программа выполняет следующие действия:

1. Создает несколько потоков для параллельной загрузки веб-страниц с разных URL-адресов.
2. Каждый поток использует библиотеку **libcurl** для выполнения HTTP-запроса и загрузки содержимого веб-страницы.
3. В случае успешной загрузки программа создает файлы.html с указанных страниц.
4. Если произошла ошибка при загрузке (например, недоступность сервера), программа выводит сообщение об ошибке.

### Компиляция программы

Процесс разработки велся на операционной системе Linux Fedora, поэтому и исполняемый файл сделан для системы Linux.

Для успешной компиляции:

1. Скачивается библиотека libcurl: sudo dnf install libcurl-devel
2. И затем происходит компиляция с помощью команды: g++ -o my\_program main.cpp -lcurl

# Код

#include <iostream>

#include <vector>

#include <thread>

#include <curl/curl.h>

#include <fstream>

// Функция для загрузки веб-страницы

size\_t writeCallback(void\* contents, size\_t size, size\_t nmemb, std::string\* output) {

    size\_t totalSize = size \* nmemb;

    output->append(static\_cast<char\*>(contents), totalSize);

    return totalSize;

}

// Функция, которую выполняет каждый поток

void downloadWebPage(const std::string& url, const std::string& filename) {

    CURL\* curl = curl\_easy\_init();

    if (curl) {

        curl\_easy\_setopt(curl, CURLOPT\_URL, url.c\_str());

        // Буфер для хранения загруженной веб-страницы

        std::string buffer;

        // Установка колбека для записи данных в буфер

        curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, writeCallback);

        curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, &buffer);

        // Выполнение запроса

        CURLcode res = curl\_easy\_perform(curl);

        // Обработка ошибок

        if (res != CURLE\_OK) {

            std::cerr << "Error downloading " << url << ": " << curl\_easy\_strerror(res) << std::endl;

        } else {

            // Сохранение в буфере в файл

            std::ofstream outputFile(filename);

            if (outputFile.is\_open()) {

                outputFile << buffer;

                std::cout << "Downloaded content from " << url << " saved to " << filename << std::endl;

            } else {

                std::cerr << "Error opening file " << filename << " for writing." << std::endl;

            }

        }

        // Освобождение ресурсов CURL

        curl\_easy\_cleanup(curl);

    }

}

int main() {

    // Список URL-адресов для загрузки

    std::vector<std::string> urls = {};

    // Пополение адресов

    std::cout << "Введите url адреса через enter, когда закончите, введите 0: " << std::endl;

    std::string url;

    do {

        std::cin >> url;

        urls.push\_back(url);

    } while (url != "0");

urls.pop\_back();

    // Вектор для хранения потоков

    std::vector<std::thread> threads;

    // Запуск потоков для загрузки веб-страниц

    for (size\_t i = 0; i < urls.size(); ++i) {

        threads.emplace\_back(downloadWebPage, urls[i], "output" + std::to\_string(i) + ".html");

    }

    // Дождитесь завершения всех потоков

    for (auto& thread : threads) {

        thread.join();

    }

    return 0;

}

# Тестирование

Сделаем тесты (таблица 1).

Таблица 1 - тесты

| Номер теста | Описание теста | Входные данные | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Загрузка HTML с корректного веб-сайта | URL корректного сайта | Успешная загрузка HTML |
| 2 | Загрузка HTML с некорректного веб-сайта | URL некорректного сайта | Ошибка при загрузке HTML |
| 3 | Сохранение HTML в файл | HTML-данные, путь для сохранения | Файл создан и содержит HTML |
| 4 | Парсинг корректного HTML | HTML с верной структурой | Структура данных построена успешно |
| 5 | Парсинг HTML с неверной структурой | HTML с некорректной структурой | Обработка ошибок при разборе HTML |
| 6 | Параллельная загрузка данных с нескольких страниц | Список URL для загрузки | Все страницы успешно загружены |
| 7 | Обработка ошибок ввода-вывода | Попытка чтения/записи в недоступный файл | Обработка ошибок ввода-вывода |
| 8 | Обработка ошибок ввода HTML | HTML с ошибками | Обработка ошибок в разборе HTML |

И проверим их покрытие по черному ящику (таблица 2).

Таблица 2 - ЧЯ

| Функциональные модули | Всего тестов | Успешно пройдено | Неудачно пройдено | Покрытие (%) |
| --- | --- | --- | --- | --- |
| Модуль ввода-вывода | 3 | 3 | 0 | 100 |
| Модуль разбора HTML | 5 | 4 | 1 | 80 |
| Многопоточность | 1 | 1 | 0 | 100 |
| Обработка ошибок | 3 | 2 | 1 | 66.67 |
| Общее покрытие | 12 | 10 | 2 | 83.33 |

# Приложение А. Скриншоты работы программы

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рисунок - Пример работы