­­Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ЛАБОРАТОРНАЯ РАБОТА № 4

по курсу: «Алгоритмизация и программирование»

на тему:

**«Разработка приложения на С++ в объектном стиле»**

Выполнил

студент группы КТбо1-2 И. Н. Акимов

Принял

Доцент В. С. Лапшин

Таганрог 2024

**Техническое задание**

**Задание**

Разработать классы для реализации индивидуального задания и отладить их в консольном приложении для Windows, используя библиотеку .NET.

Вариант 2

Что дано Что получить

|  |  |
| --- | --- |
| Список адресов сайтов  Интернет | Набор сайтов, размещённых либо на одном сервере, либо в одном каталоге сервера, либо использующих одинаковый протокол |

**Ход работы**

При запуске вносятся адреса сайтов.

После адреса разделяются на протокол, домен и путь.

Далее пользователю предлагается выбрать критерий группировки.

После чего выводятся адреса сайтов, разделенные по группам.

**Описание кода**

1. **Website** - класс представляет сайт с URL, протоколом, доменом и путем. Метод ParseUrl использует регулярное выражение для разбора URL.
2. **WebsiteGroup** - класс с методами для группировки сайтов по домену, протоколу и пути.
3. **main** - основной метод, который содержит предопределённый список URL-адресов:
   * Предлагает выбор критерия группировки.
   * Выполняет группировку и выводит результаты.

**Объяснение разбора URL**

* **Регулярное выражение**: std::regex url\_regex(R"((http|https)://([^/]+)(/?.\*))");
  + (http|https) - протокол (HTTP или HTTPS).
  + :// - разделитель протокола и домена.
  + ([^/]+) - домен, состоящий из любых символов, кроме слеша.
  + (/?.\*) - путь, который может быть пустым или содержать любые символы.
* **std::smatch**: используется для хранения результатов сопоставления.
* **std::regex\_match**: проверяет, соответствует ли URL регулярному выражению, и извлекает части URL.

**Пример работы программы**

****

*Рисунок 1 – Пример работы программы*

**Вывод**

Проделав эту лабораторную работу, я узнал и освоил возможности языка С++ связанные с работой с классами и принцип работы ООП в C++.

**Исходный код**

4laba4

#include "pch.h"

#include "Header.h"

int main() {

setlocale(LC\_ALL, "Russian");

vector<Website> websites = {

Website("https://example.com"),

Website("https://example.com/about"),

Website("http://example.net"),

Website("https://example.net/contact"),

Website("https://blog.example.com"),

Website("https://shop.example.com"),

Website("http://another-example.com"),

Website("https://another-example.com/products"),

Website("https://another-example.com/services")

};

cout << "Выберите критерий группировки:" << endl;

cout << "1. По домену" << endl;

cout << "2. По протоколу" << endl;

cout << "3. По пути" << endl;

int choice;

cin >> choice;

map<string, vector<Website>> groupedWebsites;

switch (choice) {

case 1:

groupedWebsites = WebsiteGroup::GroupByDomain(websites);

break;

case 2:

groupedWebsites = WebsiteGroup::GroupByProtocol(websites);

break;

case 3:

groupedWebsites = WebsiteGroup::GroupByPath(websites);

break;

default:

cout << "Некорректный выбор." << endl;

main();

}

// Вывод результатов

for (const auto& group : groupedWebsites) {

cout << "\nКлюч: " << group.first << endl;

for (const auto& website : group.second) {

cout << " - " << website.Url << endl;

}

}

return 0;

}

Header.h

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <map>

#include <regex>

using namespace System;

using namespace std;

// Класс для представления сайта

class Website {

public:

string Url;

string Protocol;

string Domain;

string Path;

Website(const string& url) : Url(url) {

ParseUrl();

}

void ParseUrl() {

regex url\_regex(R"((http|https)://([^/]+)(/?.\*))");

smatch url\_match\_result;

if (regex\_match(Url, url\_match\_result, url\_regex) && url\_match\_result.size() == 4) {

Protocol = url\_match\_result[1].str();

Domain = url\_match\_result[2].str();

Path = url\_match\_result[3].str();

}

else {

Protocol = "";

Domain = "";

Path = "";

}

}

string GetDomain() const {

return Domain;

}

string GetProtocol() const {

return Protocol;

}

string GetPath() const {

return Path;

}

};

// Класс для группировки сайтов

class WebsiteGroup {

public:

static map<string, vector<Website>> GroupByDomain(const vector<Website>& websites) {

map<string, vector<Website>> grouped;

for (const auto& website : websites) {

grouped[website.GetDomain()].push\_back(website);

}

return grouped;

}

static map<string, vector<Website>> GroupByProtocol(const vector<Website>& websites) {

map<string, vector<Website>> grouped;

for (const auto& website : websites) {

grouped[website.GetProtocol()].push\_back(website);

}

return grouped;

}

static map<string, vector<Website>> GroupByPath(const vector<Website>& websites) {

map<string, vector<Website>> grouped;

for (const auto& website : websites) {

grouped[website.GetPath()].push\_back(website);

}

return grouped;

}

};