**For learning :** [**https://www.learncpp.com/**](https://www.learncpp.com/)

**https://www.cppreference.com/**

**Install Crow (Easy way for beginners)**

We'll install **Crow** using **vcpkg**, a package manager for C++ libraries.

**📦 A. Install vcpkg**

Open **PowerShell** (or CMD) and run:

bash

CopyEdit

git clone https://github.com/microsoft/vcpkg.git

cd vcpkg

.\bootstrap-vcpkg.bat

After it finishes, you’ll have a vcpkg.exe file.

**📥 B. Install Crow & its dependencies**

bash

CopyEdit

.\vcpkg.exe install crow

1. Cloned vcpkg
2. ✅ Bootstrapped it with .\bootstrap-vcpkg.bat

Now you're ready for...

**🌐 Step 4: Install the Crow C++ Web Framework**

**step 5: Create Your First Web Server (main.cpp)**

**after the server is done, make vector, struct in main.cpp for the blog entries**

**#include <crow.h>**

**#define CROW\_MAIN**

**#include "crow.h"**

**#include<vector>**

**#include <string>**

**#include <sstream>**

**struct blogpost\_struct {**

**int id;**

**std::string title;**

**std::string content;**

**};**

**std::vector<blogpost\_struct> blogpost\_vector = {**

**{1, "First Post", "This is the content of the first post."},**

**{2, "Second Post", "This is the content of the second post."},**

**{3, "Third Post", "This is the content of the third post."}**

**};**

**int main()**

**{**

**crow::SimpleApp app;**

**CROW\_ROUTE(app, "/")([](){**

**std::ostringstream html;**

**html << "<h1>Welcome to Tanseem's Blogging Platform!</h1>";**

**html << "<ul>";**

**for (const auto& post : blogpost\_vector) {**

**html << "<li><strong>" << post.id << "</strong>" << post.title << "</li><br>";**

**}**

**html << "</ul>";**

**return html.str();**

**});**

**app.port(18080).multithreaded().run();**

**}**

#define CROW\_MAIN

#include "crow.h"

#include <vector>

#include <string>

#include <sstream>

struct blogpost\_struct {

    int id;

    std::string title;

    std::string content;

};

*// Initial dummy posts*

std::vector<blogpost\_struct> blogpost\_vector = {

    {1, "First Post", "This is the content of the first post."},

    {2, "Second Post", "This is the content of the second post."},

    {3, "Third Post", "This is the content of the third post."}

};

*// Next available post ID*

int next\_id = 4;

int main()

{

    crow::SimpleApp app;

*// 🏠 Home Route - Show list of posts*

    CROW\_ROUTE(app, "/")([](){

    std::ostringstream html;

    html << "<h1>Welcome to Tanseem's Blogging Platform!</h1>";

    html << "<a href='/add'>➕ Add New Post</a><br><br>";

    html << "<ul>";

    for (const auto& post : blogpost\_vector) {

        html << "<li><strong>" << post.id << ".</strong> ";

        html << "<a href=\"/post/" << post.id << "\">" << post.title << "</a>";

        html << " - <a href='/delete/" << post.id << "' style='color:red;'>Delete</a></li><br>";

    }

    html << "</ul>";

    return html.str();

});

*//add route*

*/\**

*CROW\_ROUTE(app, "/add")([](){*

*std::ostringstream html;*

*html << "<h1>Add a New Blog Post</h1>";*

*html << "<form action='/submit' method='GET'>";*

*html << "Title: <input type='text' name='title'><br><br>";*

*html << "Content:<br><textarea name='content' rows='5' cols='40'></textarea><br><br>";*

*html << "<input type='submit' value='Post'>";*

*html << "</form>";*

*html << "<br><a href='/'>Back to Home</a>";*

*return html.str();*

*});\*/*

CROW\_ROUTE(app, "/add")([](){

    std::cout << "⚠️ /add route accessed\n"; *// This line will print to terminal*

    std::ostringstream html;

    html << "<h1>Add a New Blog Post</h1>";

    html << "<form action='/submit' method='GET'>";

    html << "Title: <input type='text' name='title'><br><br>";

    html << "Content:<br><textarea name='content' rows='5' cols='40'></textarea><br><br>";

    html << "<input type='submit' value='Post'>";

    html << "</form>";

    html << "<br><a href='/'>Back to Home</a>";

    return html.str();

});

CROW\_ROUTE(app, "/submit")

([](const crow::request& *req*){

    auto title = *req*.url\_params.get("title");

    auto content = *req*.url\_params.get("content");

    if (!title || !content) {

        return crow::response(400, "Missing title or content!");

    }

    int nextId = blogpost\_vector.empty() ? 1 : blogpost\_vector.back().id + 1;

    blogpost\_struct newPost = {nextId, title, content};

    blogpost\_vector.push\_back(newPost);

    crow::response res;

    res.code = 302;

    res.set\_header("Location", "/");

    return res;

});

*/\**

*CROW\_ROUTE(app, "/submit")*

*([](const crow::request& req){*

*auto title = req.url\_params.get("title");*

*auto content = req.url\_params.get("content");*

*if (!title || !content) {*

*return "Missing title or content!";*

*}*

*// Assign the next ID*

*int nextId = blogpost\_vector.empty() ? 1 : blogpost\_vector.back().id + 1;*

*// Create a new blogpost\_struct*

*blogpost\_struct newPost = {nextId, title, content};*

*// Add to vector*

*blogpost\_vector.push\_back(newPost);*

*// Redirect to home*

*/\*return crow::response(302, "", {{"Location", "/"}});  // Redirect to /*

*});\*/*

crow::response res;

res.code = 302;

res.set\_header("Location", "/");

return res;

});\*/

*// 🗑️ Route to delete a post*

*/\**

*CROW\_ROUTE(app, "/delete/<int>")*

*([](int postId){*

*for (auto it = blogpost\_vector.begin(); it != blogpost\_vector.end(); ++it) {*

*if (it->id == postId) {*

*blogpost\_vector.erase(it);*

*break;*

*}*

*}*

*return crow::response(302, "", {{"Location", "/"}}); // Redirect to homepage*

*});\*/*

CROW\_ROUTE(app, "/delete/<int>")

([](int *postId*){

    for (auto it = blogpost\_vector.begin(); it != blogpost\_vector.end(); ++it) {

        if (it->id == postId) {

            blogpost\_vector.erase(it);

            break;

        }

    }

    crow::response res;

    res.code = 302;

    res.set\_header("Location", "/");

    return res;

});

*// 📄 Route to view individual post*

    CROW\_ROUTE(app, "/post/<int>")([](int *id*){

        for (const auto& post : blogpost\_vector) {

            if (post.id == id) {

                std::ostringstream html;

                html << "<h2>" << post.title << "</h2>";

                html << "<p>" << post.content << "</p>";

                return crow::response(html.str());

            }

        }

        return crow::response(404, "Post not found.");

    });

*/\* // 📝 Route to submit a new post via POST*

*CROW\_ROUTE(app, "/submit").methods("POST"\_method)([](const crow::request& req){*

*auto body = crow::json::load(req.body);*

*if (!body) {*

*return crow::response(400, "Invalid JSON");*

*}*

*std::string title = body["title"].s();*

*std::string content = body["content"].s();*

*blogpost\_vector.push\_back({next\_id++, title, content});*

*return crow::response(200, "Post added successfully.");*

*});\*/*

*// 🔄 Route to get all posts in JSON*

*/\*CROW\_ROUTE(app, "/posts")([](){*

*crow::json::wvalue result;*

*result["posts"] = crow::json::wvalue::list();*

*for (const auto& post : blogpost\_vector) {*

*crow::json::wvalue item;*

*item["id"] = post.id;*

*item["title"] = post.title;*

*item["content"] = post.content;*

*result["posts"].push\_back(item);*

*}*

*return crow::response(result);*

*});\*/*

    CROW\_ROUTE(app, "/posts")([](){

    crow::json::wvalue result;

    std::vector<crow::json::wvalue> post\_list;

    for (const auto& post : blogpost\_vector) {

        crow::json::wvalue item;

        item["id"] = post.id;

        item["title"] = post.title;

        item["content"] = post.content;

        post\_list.push\_back(item);

    }

    result["posts"] = std::move(post\_list);

    return crow::response(result);

});

*//CROW\_ROUTE(app, "/delete")*

    app.port(18080).multithreaded().run();

}