# 7. C++介紹

### 我們要寫的程式

- 程式比賽導向->解題目快速寫完
- 軟(韌)體->讓之後比較好懂不注重省資源清楚明白的程式(可能逐漸被高階語言取代)

這學期要上的重點是第一個目標是要讓各位能掌握到**競速程式**的基礎 當然這個基礎也能幫助之後C++軟體的學習

```
174 int main() {
        int n, m;
        cin >> n >> m:
        int rtn = 0:
        int v;
        int v1. v2:
        unordered map<int, int> map; // location = key, num = val;
        for (int run = 0; run < m; run++) {
         cin >> v:
           cin >> v1 >> v2:
            if (map.find(v1) != map.end() && map.find(v2) != map.end()) {swap(map[v1], map[v2]);}
            else if (map.find(v1) != map.end()) \{map[v2] = map[v1]; map[v1] = v2;\}
            else if (map.find(v2) != map.end()) \{map[v1] = map[v2] : map[v2] = v1:\}
            else \{map[v1] = v2; map[v2] = v1;\}
          } else if (v == 2) {
            cin >> v1:
            if (map.find(v1) != map.end()) {rtn += map[v1];}
           else {rtn += v1;}
        cout << rtn << endl:
        return 0:
```

```
Project ~
                                       @ server_util.cpp >
                                         42 \( \sqrt{void session(int client_fd, sockaddr_in client_addr) \}

∨ I Tunnel ~/Projects/Tunnel

                                                  std::atomic<bool> echo_heartbeat ( i: false);
   > 🗀 .vscode
                                                  std::atomic<bool> close_session_flag ( i: false);
   > 🗀 bin

∨ □ client

                                                  fd_set read_fds;
        client_config.cpp
                                                  int ready_for_call;
        dictiont_config.hpp
                                                  timeval timev;
        client_util.cpp
        il client_util.hpp
                                                  int nbytes = 0;

∨ □ common

                                                 char inbuffer[1024] = {0}, outbuffer[1024] = {0};
        the message.cpp
                                                  std::thread heartbeat_thread( &: heartbeat, std::ref( &: cl:
        message.hpp
        tandom_key.cpp
                                                  std::thread stream thread:
        in random_kev.hpp
                                                  std::unordered_map<int, int> user_socket_fds( n: 32); // u
   > m demo service

∨ □ server

                                                  int new_port = 0;
        server_config.cpp
                                                  std::atomic<bool> port_connected( |: false);
        iserver_config.hpp
        server_util.cpp
                                                  Message stream_port_message;
        server_util.hpp
                                                  stream_port_message.type = STREAM_PORT;
      aitianore .
      client main.cpp
```

### 最最最基本的C++程式架構

可參照 basic.cpp

```
#include <iostream> // 標頭檔 取得電腦需要的code
   using namespace std; // 將std的所有成員提到全域 (看不懂的話就把他當魔法)
   例如: std::string 可以直接用 string
#include <path to header> or #include "path to header"
告訴編譯器這個程式需要哪些功能 (對應的"標頭檔 header")
using namespace name of namespace;
*競速程式常用的做法 原本大部分的功能都被包在 "std" 的空間裡面
會讓原本 std底下的所有函式跑到全域
例如:
using namespace std;
                               using namespace std::chrono;
std::string() 會變成 string()
                               std::chrono::milliseconds 會變成 milliseconds
```

header <cmath> or < math.h> "~/test/test.hpp" (string) cos () 4117 string() 5 in () frll append1) tan() insert() size() test. CPP # include (string> # include (cmath) 妹include "n/test/test.hpp"

## 最最最基本的C++程式架構

```
註解
就是讓程式更好懂的東西
單行註解用 / / 開頭
// 這後面的所有東西都是註解
多行註解用 /*和 */ 包起來
/*
這中間
所有東西
都是註解
*/
```

```
// 這是註解
/* ← 這個是註解的開頭
這是貓貓
註解到這裡結束 → */
```

#### 最最最基本的C++程式架構

C++的程式由上到下, 左到右執行

所有程式都由主程式開始

主程式就是main() {}