

物件導向小專題 快問快答

班級：資工1B 組別：第五組

組員：B3231165 李威德、B3231181 吳浚愷

B3231149 郭成愷

專題目標

本專題旨在運用物件導向程式設計（OOP）的核心觀念，使用 C++ 語言開發一款具有趣味性與挑戰性的限時答題小遊戲「快問快答」。透過題庫管理、答題計分、時間限制等模組的規劃與設計，訓練團隊在軟體開發中模組化思考、程式架構規劃、以及實際協作的的能力。

本遊戲適合作為程式邏輯練習、基礎測驗或娛樂使用，遊戲簡單直觀，但可延伸性高，例如未來可加入GUI介面、題庫外部讀取、難度等級調整、排行榜等功能。





架構設計

1. 題庫系統 (Question Module)

透過 Question 類別封裝每一題的問題內容、選項、與正確答案索引。

2. 計分系統 (Score System)

每答對一題可獲得固定分數 (如 10 分)。

根據最終得分給予評等 (如「幹得不錯」)。

3. 時間控制系統 (Time Limit System)

每題限時作答，例如 15 秒。

使用 C++11 的 `<chrono>` 與 `<thread>` 標準函式庫來模擬倒數與非同步計時。

相關技術

在本專案中，我們使用class來設計題目（Question）與遊戲管理（GameManager）等模組，將程式邏輯與資料封裝於物件中，使整體架構更清晰、擴充性更佳。

運用的 OOP 概念：

抽象（Abstraction）

類別（Class）與物件（Object）

責任分工與模組化設計

封裝（Encapsulation）

遊戲進行流程

1. 遊戲開始 (Game Start)

顯示歡迎訊息與遊戲說明，玩家按任意鍵開始遊戲。

提醒玩家：每一題限時作答（例如：15 秒內作答完畢）。

2. 顯示題目與倒數計時 (Question Display & Countdown)

系統隨機選取或依序取出一題題目，並列出 4 個選項。

顯示倒數計時，玩家需在時間內輸入答案（1~4）。

3. 玩家作答 (Player Input)

玩家輸入答案（數字1~4），系統接收輸入，若在時間內輸入成功則進入下一步。

若未在時間內輸入，系統視為逾時不作答，直接提示「時間到，跳過本題」。

遊戲進行流程

4.判斷正確或錯誤 (Answer Evaluation)

系統比對玩家輸入的答案與正確答案索引。

若答對：加分（如 +10 分），並顯示「答對了！」

若答錯：顯示「答錯了！正確答案是 ○○」。

若逾時：顯示「時間到！此題未作答」。

5.顯示下一題或結算結果 (Next Question / Game End)

若還有剩餘題目，繼續回到步驟 2。

若所有題目作答完畢，顯示最終得分與評等，例如：

50 分以上 → 大神求抱大腿。

30~49 分 → 唉叻不錯叻。

0~29 分 → 菜就多練。

顯示結束畫面與感謝遊玩訊息。



main.cpp

這段是程式的主函式 `main()`，功能很簡單：

建立一個 `GameManager` 物件。

呼叫 `start()` 開始遊戲流程。

遊戲結束後正常回傳 0，表示程式順利結束。

整個程式從這裡開始執行，帶動問答遊戲的運作～

```
1      #include "GameManager.h"
2
3  ✓  int main() {
4      GameManager game;
5      game.start();
6      return 0;
7  }
```

Question.h

這段程式碼定義了一個 Question 類別，包含問題文字、選項陣列與正確答案的索引。提供建構子建立問題，display 用來顯示題目，isCorrect 檢查使用者的答案是否正確。設計簡潔，適合用於小測驗系統。

```
1  #ifndef QUESTION_H
2  #define QUESTION_H
3
4  #include <string>
5  #include <vector>
6
7  class Question {
8  private:
9      std::string text;
10     std::vector<std::string> options;
11     int correctIndex;
12
13 public:
14     Question(std::string t, std::vector<std::string> opts, int correct);
15     void display() const;
16     bool isCorrect(int choice) const;
17 };
18
19 #endif
```


Question.cpp

建構子用初始化列表設定問題文字、選項與正確答案位置；display 方法印出問題與選項；isCorrect 檢查使用者輸入的選項是否等於正確答案的編號（從 1 開始）。整體簡潔直觀，適合小測驗功能使用。

```
1  #include "Question.h"
2  #include <iostream>
3
4  using namespace std;
5
6  Question::Question(string t, vector<string> opts, int correct)
7      : text(t), options(opts), correctIndex(correct) {}
8
9  void Question::display() const {
10     cout << text << endl;
11     for (size_t i = 0; i < options.size(); ++i) {
12         cout << i + 1 << ". " << options[i] << endl;
13     }
14 }
15
16 bool Question::isCorrect(int choice) const {
17     return choice == correctIndex + 1;
18 }
```

GameManager.h

這段程式碼定義了 GameManager 類別，用來管理整個問答遊戲流程。它包含一個問題列表、一個分數計數器，以及一個固定的時間限制。提供建構子初始化遊戲、start() 開始問答流程，並有私有方法 getUserAnswerWithTimer() 處理限時作答、showRank() 顯示排名。整體設計清楚，有利於遊戲流程控制。

```
1  #ifndef GAMEMANAGER_H
2  #define GAMEMANAGER_H
3
4  #include <vector>
5  #include "Question.h"
6
7  class GameManager {
8  private:
9      std::vector<Question> questions;
10     int score;
11     const int timeLimit;
12
13 public:
14     GameManager();
15     void start();
16
17 private:
18     int getUserAnswerWithTimer();
19     void showRank();
20 };
21
22 #endif
```

GameManager.cpp

這段是 GameManager 類別的實作，功能說明如下：
建構子初始化分數為 0、時間限制為 15 秒，並建立 5 題選擇題。

start() 方法開始遊戲，依序顯示每題並等待玩家作答。
若超時則不計分，答對加 10 分，答錯會顯示正確答案。

```
34     }
35 }
36
37 cout << "遊戲結束！你的總分是：" << score << " 分。\\n";
38 showRank();
39 }
40
41 int GameManager::getUserAnswerWithTimer() {
42     int choice = -1;
43     auto startTime = steady_clock::now();
44     bool inputReceived = false;
45
46     thread inputThread([&]() {
47         cout << "請輸入你的答案（數字1-4）：";
48         cin >> choice;
49         inputReceived = true;
50     });
51
52     while (duration_cast<seconds>(steady_clock::now() - startTime).count() < timeLimit) {
53         if (inputReceived) break;
54         this_thread::sleep_for(milliseconds(100));
55     }
56
57     if (!inputReceived) {
58         inputThread.detach();
59         return -1;
60     } else {
61         inputThread.join();
62         return choice;
63     }
64 }
```

GameManager.cpp

start() 結尾印出總分，並呼叫 showRank() 顯示排名。

getUserAnswerWithTimer() 使用多執行緒監聽使用者輸入，並搭配時間限制判斷是否超時。

在新執行緒讀取輸入，主執行緒則持續檢查是否有輸入或超過時間。若時間到沒輸入，分離輸入執行緒並回傳 -1 表示超時。

若有輸入，等待輸入執行緒結束後回傳答案。

```
34     }
35 }
36
37     cout << "遊戲結束！你的總分是：" << score << " 分。\\n";
38     showRank();
39 }
40
41 int GameManager::getUserAnswerWithTimer() {
42     int choice = -1;
43     auto startTime = steady_clock::now();
44     bool inputReceived = false;
45
46     thread inputThread([&]() {
47         cout << "請輸入你的答案（數字1-4）： ";
48         cin >> choice;
49         inputReceived = true;
50     });
51
52     while (duration_cast<seconds>(steady_clock::now() - startTime).count() < timeLimit) {
53         if (inputReceived) break;
54         this_thread::sleep_for(milliseconds(100));
55     }
56
57     if (!inputReceived) {
58         inputThread.detach();
59         return -1;
60     } else {
61         inputThread.join();
62         return choice;
63     }
64 }
```

GameManager.cpp

這段 showRank() 方法會根據玩家分數顯示不同的評語：

分數 40 分以上，稱讚為「大神求抱大腿」。

20 分以上但不到 40 分，顯示「唉叻不錯叻」。

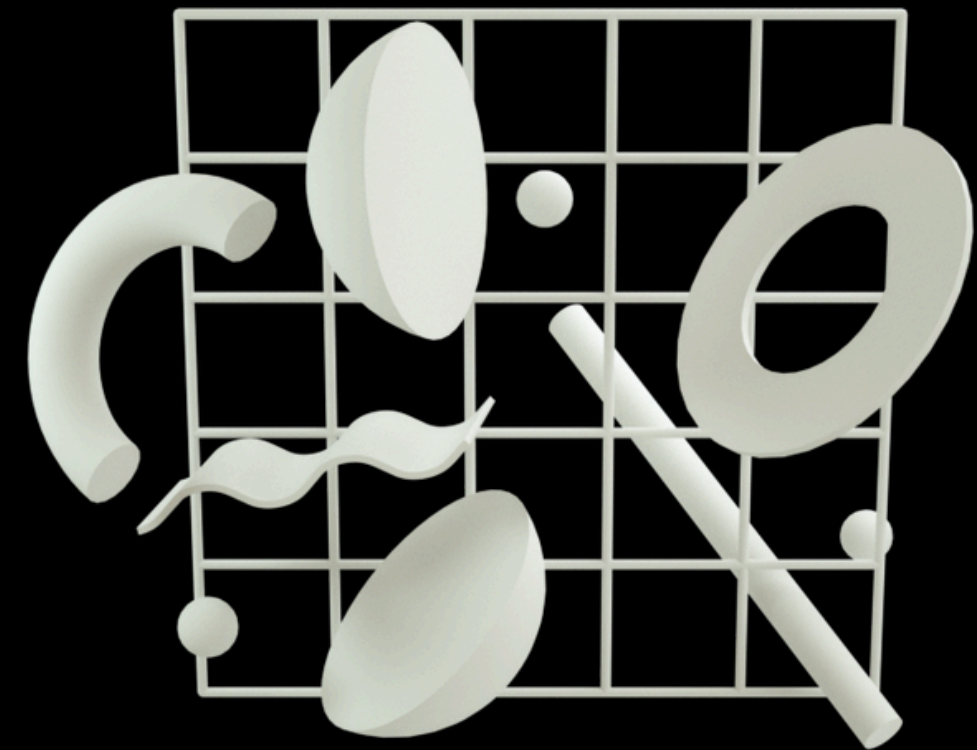
低於 20 分，鼓勵「菜就多練」。

簡單直接，用輕鬆幽默的語氣增加遊戲趣味感～

```
66  void GameManager::showRank() {  
67      if (score >= 40) {  
68          cout << "大神求抱大腿。";  
69      } else if (score >= 20) {  
70          cout << "唉叻不錯叻。\\n";  
71      } else {  
72          cout << "菜就多練。\\n";  
73      }  
74  }
```


執行方式（以 Code::Blocks 為例）

1. 開啟 Code::Blocks，選擇 `File > New > Project > Console Application`。
2. 選擇 C++，按「Next」，設定專案名稱與儲存位置。
3. 在左側 Project 樹中右鍵點選「Sources」→「Add files」→ 加入「main.cpp」、「GameManager.cpp」、「Question.cpp」。
4. 同理加入「Headers」→「GameManager.h」、「Question.h」。
5. 點選「Build and Run (F9)」開始執行。



UML 類別圖

這張 UML 類別圖包含兩個主要類別：

Question 類別：負責儲存單一題目的內容與判斷邏輯

GameManager 類別：
管理整個遊戲流程、題庫、計分與時間控制

兩者之間的關係為：

GameManager 使用（uses）Question 類別作為其題庫資料的構成元素。

```
+-----+
|   Question   |
+-----+
| - text: string |
| - options: vector<string> |
| - correctIndex: int |
+-----+
| + display(): void |
| + isCorrect(choice: int): bool |
+-----+

      ▲
      |
      | uses |
      |
+-----+
|   GameManager   |
+-----+
| - questions: vector<Question> |
| - score: int |
| - timeLimit: int |
+-----+
| + GameManager() |
| + start(): void |
| - getUserAnswerWithTimer(): int |
| - showRank(): void |
+-----+
```

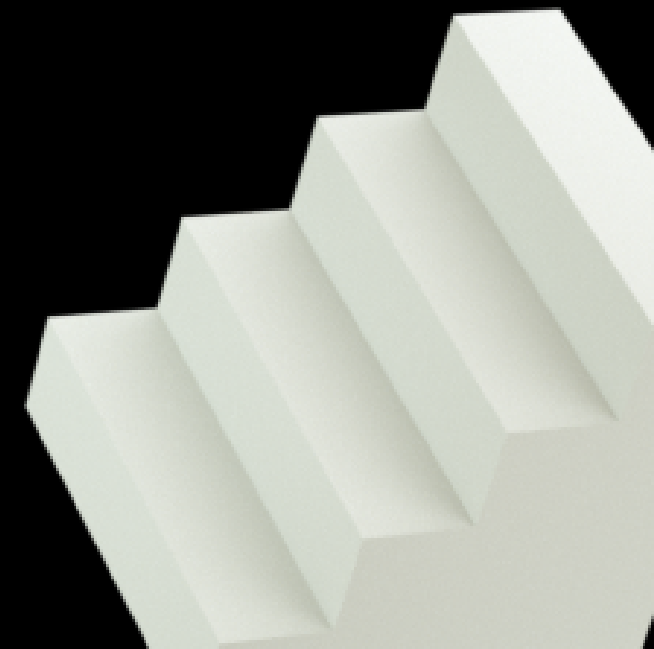
程式運行畫面截圖





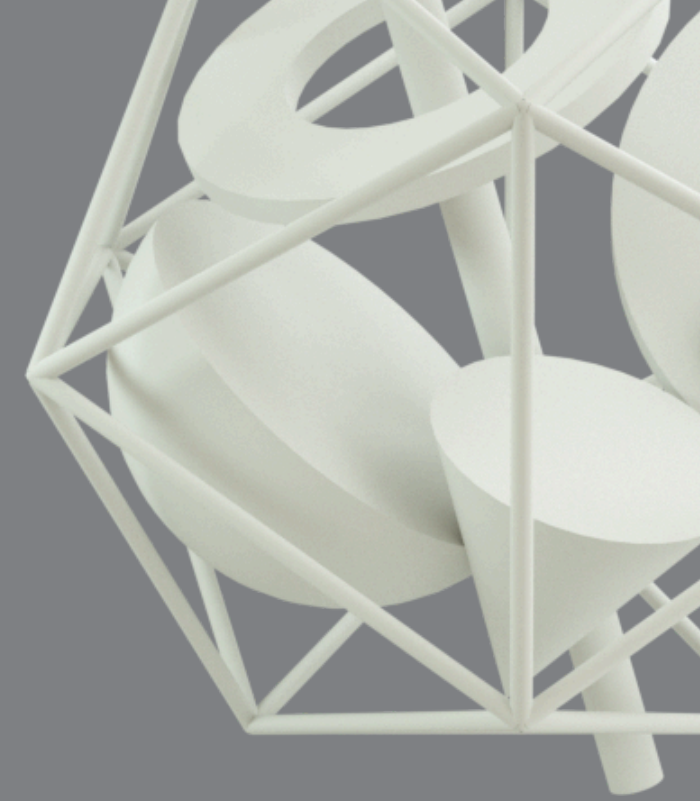
小結與心得

這次專題讓我們學會了物件導向設計的核心思維，懂得如何將功能模組化並建立良好結構；在開發過程中，也體會到團隊合作與明確分工對專案完成的關鍵性。我們成功解決了倒數計時與使用者輸入同步的技術挑戰，讓遊戲能準確判定答題時限。未來可以擴充更多題庫資料，並增加不同挑戰模式，如雙人對戰或難度分級，使遊戲更豐富多元。這次專題讓我們對軟體開發有了更完整的實作經驗。

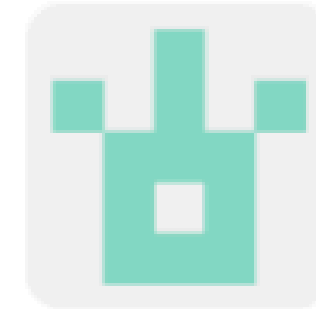




github



FanQiLin9487/-



1
Contributor

0
Issues

0
Stars

0
Forks



FanQiLin9487/

Contribute to FanQiLin9487/- development by creating an account on GitHub.

 GitHub

<https://github.com/FanQiLin9487/->



成員分工

李威德：題庫生成、流程圖製作

吳浚愷：介面設計、UML繪製

郭成愷：計分與時間控制、

GitHub整理

