PRG155 Lab 8：for 迴圈 — 詳細解題報告

溫度讀取程式（Device Alpha 開發板）

繳交期限：2026 年 7 月 23 日實驗課開始前 ｜ 繳交檔案：`lab8.c` ｜ 配分：2 分

---

1 題目說明

本實驗要求撰寫一支 C 程式，透過 **Device Alpha** 實驗板（LED、蜂鳴器、加熱器、溫度感測器、七段顯示器、按鈕）完成一個「多次溫度讀取並計算平均值」的流程。題目的完整要求如下：

1. 程式開頭必須包含指定格式的註解（程式名稱、作者、日期、目的）。
2. 程式流程：
   (a) 點亮**藍色 LED**，詢問使用者要讀取幾次溫度。
   (b) 取得有效輸入後：關閉藍色 LED、開啟**加熱器（HEATER）**、點亮**綠色 LED**。
   (c) 每一次讀取：先在**七段顯示器**顯示目前讀取編號（1、2、3、…），**暫停 100 毫秒**後才讀取溫度（讓顯示器更新、感測器穩定）。
   (d) 兩次讀取之間**暫停 1 秒**；暫停期間關閉綠色 LED、點亮**黃色 LED**；暫停結束後關閉黃色 LED、重新點亮綠色 LED。
   (e) 完成指定次數後：關閉加熱器，計算並回報**平均溫度**（溫度須使用適當型別 `double`），並將平均溫度以**小數點後 2 位**顯示在七段顯示器上。
   (f) 程式結束前，必須等待使用者**按下任意按鈕並放開**後才結束。
3. 解法中**必須使用 for 迴圈**。
4. **輸入驗證**：若使用者輸入無效值（非數字、0 或負數），必須顯示「Not a valid amount!」訊息，開啟**蜂鳴器（BUZZER）**與**紅色 LED**，等待使用者按下並放開任意按鈕後，關閉蜂鳴器與紅色 LED，再重新詢問一次。
5. 程式結構須正確使用 Tab 縮排，變數命名須使用具描述性的**小駝峰式（camelCase）**名稱，此部分列入評分。
6. 檔案命名為 `lab8.c`，上傳至 Blackboard。

題目提供的 Device Alpha 函式

```
DisplayIntValue(intVar);                /* 七段顯示器顯示整數          */
SetHeaterState(HEATER, ON);             /* 加熱器開                    */
SetHeaterState(HEATER, OFF);            /* 加熱器關                    */
double temperature;
temperature = GetTemperature(TEMPERATURE_SENSOR);   /* 讀取溫度 */
DisplayDoubleValue(varDouble, 2);       /* 顯示浮點數，指定小數位數    */
```

另外常用的還有 `SetLEDState(顏色, ON/OFF)`、`SetBuzzerState(BUZZER, ON/OFF)`、`GetPushButtonState(ANY_PUSHBUTTON)`（回傳 `PRESSED` 或 `RELEASED`）與 `Wait(毫秒)`。

為什麼要等待 100 ms 與 1 秒？(題目附註)

讀取前等待 100 ms：讓七段顯示器來得及更新讀取編號，也給溫度感測器短暫的穩定時間；若沒有這個延遲，讀取可能太快，造成顯示器閃爍或感測值稍微不準。

每次讀取後等待 1 秒：在兩次讀取之間建立明顯的停頓，讓使用者能看到 LED 切換、以及讀取進度以穩定的節奏前進；若沒有這個延遲，整個程式幾乎瞬間跑完，使用者看不到每個步驟。

2 問題分析

2.1 硬體裝置與其角色

裝置	何時使用	代表意義
藍色 LED	程式開始、等待輸入時	「待機 / 等待輸入」狀態
綠色 LED	讀取溫度期間	「量測進行中」狀態
黃色 LED	兩次讀取之間的 1 秒暫停	「暫停中」狀態
紅色 LED + 蜂鳴器	輸入無效時	「錯誤警報」狀態
加熱器	整個量測期間	受測的熱源
七段顯示器	顯示讀取編號與平均溫度	使用者回饋
按鈕	錯誤確認、程式結束前	使用者確認動作

2.2 程式流程

整個程式可以拆成三大階段：

階段一（輸入與驗證）：藍色 LED 亮 → 詢問次數 → 若輸入無效（非數字 / 0 / 負數）則警報、等按鈕、重問，直到取得正整數。

階段二（量測迴圈，必須用 for）：藍燈滅、加熱器開、綠燈亮 → 對第 i 次讀取（$i = 1 \ldots n$）：七段顯示 i → 等 100 ms → 讀溫度並累加 → 若還有下一次，切換綠燈 → 黃燈、等 1 秒、再切回綠燈。

階段三（結果與結束）：加熱器關 → 計算平均值 = 總和 ÷ 次數 → 以 2 位小數顯示 → 等待按鈕「按下再放開」→ 程式結束。

2.3 幾個關鍵的設計決策

1. **溫度使用 `double` 型別**：溫度是連續量，且平均值需要小數點後 2 位，整數型別會流失精度，故總和、單次讀值、平均值皆宣告為 `double`。
2. **如何偵測「非數字」輸入**：`scanf("%d", &n)` 的回傳值代表成功讀入的項目數。輸入 `FIFTY` 時回傳 0，代表轉換失敗；此時字元仍留在輸入緩衝區內，必須用 `getchar()` 迴圈把整行清空，否則下一次 `scanf` 會立刻再失敗而造成無窮迴圈。
3. **驗證條件**：`inputStatus != 1`（非數字）或 `numberOfReadings <= 0`（0 或負數），兩者任一成立就是無效輸入。
4. **「按下並放開」的寫法**：需要兩個連續的等待迴圈——第一個迴圈在按鈕為 `RELEASED` 時空轉（等待按下），第二個迴圈在按鈕為 `PRESSED` 時空轉（等待放開）。只寫一個迴圈無法保證使用者已放開按鈕。

5. 黃燈暫停只出現在「兩次讀取之間」：題目說 “Between each temperature reading”，因此最後一次讀取之後不需要再暫停 1 秒，用 `if (readingNumber < numberOfReadings)` 判斷即可。

3 逐步解題過程

3.1 步驟 1：檔頭註解與變數宣告

依題目規定放上檔頭註解，並用小駝峰式命名宣告變數：

```
int    numberOfReadings = 0;    /* 使用者要讀取的次數        */
int    readingNumber;           /* for 迴圈計數器 1,2,3,... */
int    inputStatus;             /* scanf 的回傳值            */
double currentTemperature;      /* 單次溫度讀值              */
double totalTemperature = 0.0;  /* 溫度總和（累加用）        */
double averageTemperature;      /* 平均溫度                  */
```

注意 `totalTemperature` 一定要初始化為 `0.0`，否則累加結果是垃圾值。

3.2 步驟 2：點亮藍燈並取得有效輸入

```
SetLEDState(BLUE_LED, ON);
printf("How many temperature readings will be taken? ");
inputStatus = scanf("%d", &numberOfReadings);

while (inputStatus != 1 || numberOfReadings <= 0)
{
    while (getchar() != '\n')
        ;                               /* 清空輸入緩衝區 */
    printf("Not a valid amount!\n");

    SetBuzzerState(BUZZER, ON);         /* 警報開始 */
    SetLEDState(RED_LED, ON);
    while (GetPushButtonState(ANY_PUSHBUTTON) == RELEASED)
        ;                               /* 等待按下 */
    while (GetPushButtonState(ANY_PUSHBUTTON) == PRESSED)
        ;                               /* 等待放開 */
    SetBuzzerState(BUZZER, OFF);        /* 警報結束 */
    SetLEDState(RED_LED, OFF);

    printf("How many temperature readings will be taken? ");
    inputStatus = scanf("%d", &numberOfReadings);
}
```

這裡用 `while` 做輸入驗證是合理的（題目只要求解法中「必須用到」for 迴圈，量測主體用 for），因為重試次數事先未知，屬於「條件式迴圈」的典型場景。

3.3 步驟 3：切換到量測狀態

```
SetLEDState(BLUE_LED, OFF);
SetHeaterState(HEATER, ON);
SetLEDState(GREEN_LED, ON);
```

3.4 步驟 4：for 迴圈進行 n 次量測

這是本實驗的核心。計數器從 1 開始到 numberOfReadings，如此七段顯示器直接顯示 readingNumber 就是題目要的 1、2、3、…：

```
for (readingNumber = 1; readingNumber <= numberOfReadings; readingNumber++)
{
    DisplayIntValue(readingNumber);          /* 顯示讀取編號   */
    Wait(100);                               /* 100 ms 穩定期  */

    currentTemperature = GetTemperature(TEMPERATURE_SENSOR);
    totalTemperature = totalTemperature + currentTemperature;

    if (readingNumber < numberOfReadings)    /* 只在「之間」暫停 */
    {
        SetLEDState(GREEN_LED, OFF);
        SetLEDState(YELLOW_LED, ON);
        Wait(1000);                          /* 1 秒暫停       */
        SetLEDState(YELLOW_LED, OFF);
        SetLEDState(GREEN_LED, ON);
    }
}
```

3.5 步驟 5：計算並顯示平均溫度

```
SetHeaterState(HEATER, OFF);
SetLEDState(GREEN_LED, OFF);

averageTemperature = totalTemperature / numberOfReadings;

printf("The average temperature from %d readings is %.2f\n",
       numberOfReadings, averageTemperature);
DisplayDoubleValue(averageTemperature, 2);   /* 七段顯示 2 位小數 */
```

由於 totalTemperature 是 double，除以 int 時會自動提升為浮點除法，不會發生整數除法截斷的問題。

3.6 步驟 6：等待按鈕後結束

```
while (GetPushButtonState(ANY_PUSHBUTTON) == RELEASED)
    ;                                        /* 等待按下 */
while (GetPushButtonState(ANY_PUSHBUTTON) == PRESSED)
    ;                                        /* 等待放開 */
return 0;
```

4 完整程式碼（lab8.c）

```
/*
program: lab8.c
author: firstName lastName
date: July 21 2026
purpose: using for loop
```

```c
*/

#include <stdio.h>
#include "deviceAlpha.h"

int main(void)
{
    int numberOfReadings = 0;   /* how many readings the user wants   */
    int readingNumber;      /* for loop counter (1, 2, 3, ...)     */
    int inputStatus;        /* return value of scanf               */
    double currentTemperature;  /* one temperature reading             */
    double totalTemperature = 0.0;  /* running sum of all readings         */
    double averageTemperature;  /* totalTemperature / numberOfReadings */

    /* step 1: turn on the BLUE led and ask for the number of readings */
    SetLEDState(BLUE_LED, ON);

    printf("How many temperature readings will be taken? ");
    inputStatus = scanf("%d", &numberOfReadings);

    /* step 2: validate the input - repeat until a positive number is entered
       */
    while (inputStatus != 1 || numberOfReadings <= 0)
    {
        /* remove the invalid characters left in the input buffer */
        while (getchar() != '\n')
            ;

        printf("Not a valid amount!\n");

        /* alarm: BUZZER + RED led on, wait for press AND release of any
           button */
        SetBuzzerState(BUZZER, ON);
        SetLEDState(RED_LED, ON);

        while (GetPushButtonState(ANY_PUSHBUTTON) == RELEASED)
            ;   /* wait until a button is pressed  */
        while (GetPushButtonState(ANY_PUSHBUTTON) == PRESSED)
            ;   /* wait until the button is released */

        SetBuzzerState(BUZZER, OFF);
        SetLEDState(RED_LED, OFF);

        /* ask again */
        printf("How many temperature readings will be taken? ");
        inputStatus = scanf("%d", &numberOfReadings);
    }

    /* remove the newline left behind by the valid scanf */
    while (getchar() != '\n')
        ;

    /* step 3: BLUE off, HEATER on, GREEN on */
    SetLEDState(BLUE_LED, OFF);
    SetHeaterState(HEATER, ON);
    SetLEDState(GREEN_LED, ON);

    /* step 4: take the readings with a for loop */
```

```c
    for (readingNumber = 1; readingNumber <= numberOfReadings; readingNumber
        ++)
    {
        /* show the reading number (1, 2, 3, ...) on the 7-Segment display */
        DisplayIntValue(readingNumber);

        /* short pause so the display updates and the sensor stabilizes */
        Wait(100);

        currentTemperature = GetTemperature(TEMPERATURE_SENSOR);
        totalTemperature = totalTemperature + currentTemperature;

        /* 1 second pause BETWEEN readings: GREEN off, YELLOW on */
        if (readingNumber < numberOfReadings)
        {
            SetLEDState(GREEN_LED, OFF);
            SetLEDState(YELLOW_LED, ON);

            Wait(1000);

            SetLEDState(YELLOW_LED, OFF);
            SetLEDState(GREEN_LED, ON);
        }
    }

    /* step 5: heater off and report the average temperature */
    SetHeaterState(HEATER, OFF);
    SetLEDState(GREEN_LED, OFF);

    averageTemperature = totalTemperature / numberOfReadings;

    printf("The average temperature from %d readings is %.2f\n",
           numberOfReadings, averageTemperature);
    DisplayDoubleValue(averageTemperature, 2);

    /* step 6: wait for the user to press and release ANY button, then end */
    while (GetPushButtonState(ANY_PUSHBUTTON) == RELEASED)
        ;   /* wait for the press   */
    while (GetPushButtonState(ANY_PUSHBUTTON) == PRESSED)
        ;   /* wait for the release */

    return 0;
}
```

Listing 1: lab8.c 完整原始碼

5 如何編譯與執行

5.1 在學校的 Device Alpha 環境（正式繳交）

1. 在實驗室電腦開啟 Device Alpha 開發環境，建立新專案並將 `lab8.c` 加入專案（或直接貼上程式碼）。
2. 確認專案已包含課程提供的官方 `deviceAlpha.h`（本文件附的模擬用標頭檔不要一起繳交）。

3. 編譯並執行，依照畫面提示輸入讀取次數，觀察 LED、加熱器、七段顯示器的動作是否符合題目要求。
4. 測試無效輸入：輸入 `FIFTY`、`0`、`-5`，確認會顯示錯誤訊息、蜂鳴器與紅燈亮起、按下並放開按鈕後重新詢問。
5. 確認無誤後，將 `lab8.c`（記得把檔頭註解的姓名與日期改成自己的）上傳到 Blackboard 的 Assignments 資料夾。

5.2 在一般電腦上測試（使用附帶的模擬標頭檔）

沒有實驗板時，可以使用本解答附的 `deviceAlpha.h` 模擬器：它把 LED / 加熱器 / 蜂鳴器的動作印成 `[SIM]` 訊息、七段顯示器印成 `[7SEG]` 訊息，溫度感測器回傳「開加熱器會逐漸升溫 + 少量雜訊」的模擬值，按鈕則會自動模擬「按下再放開」。

```
$ gcc -Wall -Wextra -std=c99 -o lab8 lab8.c
$ ./lab8
```

5.3 實際執行範例（模擬器輸出）

輸入依序為 `FIFTY`（無效）、`-3`（無效）、`3`（有效）：

```
        [SIM] BLUE LED ON
How many temperature readings will be taken? Not a valid amount!
        [SIM] BUZZER ON
        [SIM] RED LED ON
        [SIM] (waiting for a button... auto-pressing)
        [SIM] BUZZER OFF
        [SIM] RED LED OFF
How many temperature readings will be taken? Not a valid amount!
        [SIM] BUZZER ON
        [SIM] RED LED ON
        [SIM] BUZZER OFF
        [SIM] RED LED OFF
How many temperature readings will be taken? 3
        [SIM] BLUE LED OFF
        [SIM] HEATER ON
        [SIM] GREEN LED ON
        [7SEG] 1
        [SIM] GREEN LED OFF
        [SIM] YELLOW LED ON
        [SIM] YELLOW LED OFF
        [SIM] GREEN LED ON
        [7SEG] 2
        [SIM] GREEN LED OFF
        [SIM] YELLOW LED ON
        [SIM] YELLOW LED OFF
        [SIM] GREEN LED ON
        [7SEG] 3
        [SIM] HEATER OFF
        [SIM] GREEN LED OFF
The average temperature from 3 readings is 21.62
        [7SEG] 21.62
```

6 常見錯誤與評分要點

1. **忘記清空輸入緩衝區**：輸入 `FIFTY` 後若不清緩衝區，`scanf` 會不斷失敗造成無窮迴圈。這是本題最容易踩到的陷阱。
2. **用 `int` 存溫度**：平均溫度會被截斷成整數，無法顯示 2 位小數，會被扣分。
3. **只檢查非數字、漏掉 0 與負數**（或反過來）：驗證條件必須同時涵蓋兩種情況。
4. **最後一次讀取後仍暫停 1 秒**：題意是「兩次讀取之間」暫停；雖然多半不會重扣，但精確的寫法應加上 `if` 判斷。
5. **按鈕只等「按下」沒等「放開」**：題目明確要求 press *and release*，需要兩個等待迴圈。
6. **結構與命名**：記得用 Tab 縮排、描述性 camelCase 變數名（如 `numberOfReadings` 而非 `n`），檔頭註解格式要與題目一致，這些都直接列入評分。
7. **沒有使用 for 迴圈**：量測主體若寫成 while，會違反題目第 3 點的硬性規定。