國立屏東大學 資訊工程系 程式設計(一)

Turnin作業8

- chapter 8
- Turnin Code: c.hw8
- Due Date: 2026/06/03 23:59 **Hard Deadline**

繳交方式說明

本次作業繳交將以資料夾的形式繳交，需要為每一題建立一個資料夾（資料夾名稱為該題題目前方之代號，第一題為□p1□□第二題為□p2□□餘以此類推），

繳交說明可參考作業3：連結

任何未依照正確繳交格式的檔案將以 0 分計算

本文使用「 」及「 」代表「空白字元」與□Enter 換行字元」，並且將使用者輸入的部份使用灰階方式顯示。另外，題目的執行結果中，如果出現「(」、「)」、「:」、「;」、「.」與「,」等符號，皆為英文半形！

P1 FunctionPointer□改編自課本15-14 到 15-16頁□Example 15-9 函式指標範例）

小明看完課本的Example 15-9 後，覺得原本的陣列資料不夠實用。將原本固定陣列修改成動態輸入資料。請同學接手小明的程式碼來完成範例中各項功能的實作，並且修改原本 **median 函式邏輯**，將函式改成將陣列排序後，取得中位數。

中位數 (Median) 說明： 請先將陣列由小到大進行排序，接著依據陣列的長度取值：

- **當陣列長度為奇數時**：直接取正中間的數字。（例如□d[]陣列長度為 5，排序後中間的數字因該為 **(1+5)/2=3**，此時請取 d[2] 的數值）。
- **當陣列長度為偶數時**：取中間兩個數字裡「前面」的那一個。（例如□d[]陣列長度為 4，排序後中間的數字因該為 **(1+4)/2=2.5**，介於 d[1] 與 d[2] 之間，此時請取 d[1] 的數值）。

因此，本題需請同學定義並實作下列函式：

- **maximum**：回傳整數陣列中的最大值。
- **minimum**：回傳整數陣列中的最小值。

- **Median**：負責將傳入的陣列資料進行排序，並依據上述說明取得並回傳中位數。
- **functionPicker**：函式指標的呼叫器，負責接收特定的分析函式（如 maximum、minimum 等）以及陣列資料，並執行傳入的函式來回傳計算結果。

請參考下面 main.c：

```
#include <stdio.h>
#include <stdlib.h>
#include "FunctionPointer.h"

int main()
{
  int *data;
  int size;

  scanf("%d", &size);

  data = (int *)malloc(size * sizeof(int));

  for (int i = 0; i < size; i++) {
      scanf("%d", &data[i]);
  }

  int num;
  num = functionPicker(maximum, data, size);
  printf("The maximum is %d.\n", num);
  num = functionPicker(minimum, data, size);
  printf("The minimum is %d.\n", num);

  num = functionPicker(Median, data, size);
  printf("The median is %d.\n", num);

  free(data);

  return 0;
}
```

待同學實作完畢後，請將所有函式的實作放入以下檔案並繳交：

- FunctionPointer.c：含有上述函式的實作。
- FunctionPointer.h：含有上述函式的宣告與定義。

另外，本題目使用以下 Makefile 進行批改：

```
all: main.c FunctionPointer.o
    gcc main.c FunctionPointer.o
```

```
FunctionPointer.o: FunctionPointer.c FunctionPointer.h
    gcc -c FunctionPointer.c
clean:
    rm -rf *.*~ *~ a.* *.o *.o~
```

本題的執行結果可參考如下：

```
[3:23▲user@ws▲p1]▲./a.out↵
1↵
999↵
The▲maximum▲is▲999.↵
The▲minimum▲is▲999.↵
The▲median▲is▲999.↵
[3:23▲user@ws▲p1]▲./a.out↵
15↵
45▲12▲89▲33▲67▲92▲15▲28▲54▲71▲8▲99▲41▲60▲22↵
The▲maximum▲is▲99.↵
The▲minimum▲is▲8.↵
The▲median▲is▲45.↵
[3:23▲user@ws▲p1]▲./a.out↵
16↵
113▲452▲73▲981▲234▲657▲12▲44▲890▲321▲566▲201▲88▲340▲710▲50↵
The▲maximum▲is▲981.↵
The▲minimum▲is▲12.↵
The▲median▲is▲234.↵
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題將以上述提供的 Makefile 進行編譯，請同學作答時 **“ 以題目提供的文件為主 ”**。
- 本題應繳交檔案如下（**main.c**、**Makefile** 則不需繳交）：
  - **FunctionPointer.c**
  - **FunctionPointer.h**

---

p2 通用型資料排序器 (Generic Bubble Sort)

在 C 語言的標準函式庫中，有一個強大的排序函式叫做 `qsort`，它可以幫任何型別的陣列進行排序。
本次作業不依賴標準函式庫，需請同學利用 **無型別指標 (void*)** 與 **函式指標 (Function Pointer)**，親自實作一個具備通用性（Generic）的氣泡排序法。
該程式在執行後，會要求使用者輸入排序方向（1 為遞增，-1 為遞減），接著程式會分別對一個「整數陣列」與一個「字串陣列」進行通用排序，並印出結果。

如何寫出「通用」的程式碼？
由於排序器不知道傳進來的陣列裝的是什麼（可能是 4 bytes 的 `int`，也可能是 8 bytes 的指標），同學在實作時必須掌握以下技巧：

1. **無型別指標 (`void*`)**：它就像是一個「萬用箱」，可以接收任何型別的位址。但缺點是**編譯器不知道它有多大**，所以你不能直接對它取值（不能寫 `*arr`），也不能直接使用陣列索引（不能寫 `arr[i]`）。
2. **指標算術 (Pointer Arithmetic)**：既然不能用 `arr[i]`，要怎麼走到第 `i` 個元素？
3. 我們必須先將 `void*` 轉型為 `char*`（因為 `char` 剛好佔 1 byte），再手動乘上每個元素的大小。
4. 公式為：元素的位址 = (char*)arr + 索引值 * 元素大小(item_size)。
5. memcpy：交換兩個未知型別的變數時，不能直接用 `=` 賦值。請先用 `malloc(item_size)` 要一塊暫存空間，然後利用 `<string.h>` 中的 `memcpy` 將資料一個 Byte 一個 Byte 進行複製交換。
6. **函式指標 (Function Pointer)**：排序器不知道怎麼比較兩個未知元素的大小，所以呼叫者必須把「比較的方法（函式）」當作參數傳入。例如 `int (*cmp)(void*, void*)` 代表傳入一個回傳整數、接收兩個 `void*` 的函式。

為了實現以上操作，本題提供 **sort.h** 來幫助同學完成本次作業，請參考以下內容：

```
#define ASCENDING 1
#define DESCENDING -1

void bubbleSort(void* arr, int size, unsigned item_size, int order, int
(*cmp)(void*, void*));
int intCompare(void* a, void* b);
int stringCompare(void* a, void* b);
void printArray(void* arr, int size, unsigned item_size, void
(*print_item)(void*));
void printInt(void* item);
void printString(void* item);
```

在提供的標頭檔中，定義了排序方向的常數，以及需要同學實作的函式宣告。本題需請同學完成以下核心函式：

- **intCompare() / stringCompare()**：這兩個函式負責比較兩個元素的大小。
- **printInt() / printString()**：負責將傳入的單一元素轉型並印出。
- **printArray()**：負責走訪陣列，並透過函式指標 **print_item** 印出陣列內的所有元素。
- **bubbleSort()**：通用型氣泡排序法的核心實作 function。

實作注意

- 在實作 **比較函式 (intCompare, stringCompare)** 時：
  1. 請先將傳入的 `void* a` 與 `void* b` **轉型**為該陣列原本的指標型別，再進行取值比較。
  2. **比較規則**：前大於後回傳正數，前小於後回傳負數，相等回傳 0。
  3. 字串陣列的元素型別為 `char*`，當我們把這個元素的「位址」傳進函式時，它會變成 **指標的指標** (char**)。
- 在實作 **printArray()** 函式時：

1. 負責走訪陣列，並透過前面提到的 **指標算術** 算出每個元素的位址。
2. 呼叫函式指標 `print_item`，把算出的位址傳進去印出。

- 在實作 **bubbleSort()** 函式時：
  1. 實作雙層迴圈的氣泡排序邏輯。利用 **指標算術** 找出相鄰兩個元素 `a` 與 `b` 的位址。
  2. 呼叫 `cmp(a, b)` 判斷是否需要交換。
  3. **資料交換**：請使用 `malloc` 宣告大小為 `item_size` 的暫存空間，並利用 `memcpy` 進行複製。

請參考下面 main.c

```
#include <stdio.h>
#include "sort.h"

int main()
{
    int arr[] = {5, 2, 9, 1, 5, 6};
    char* strarr[7] = {"Banana", "apple", "Cherry", "date", "Egg"};

    int order = 0;
    printf("Enter sort direction (1: Ascending, -1: Descending): ");
    if (scanf(" %d", &order) != 1) return 1;

    printf("\n[Before Sorting]\n");
    printf("  Integer array: [");
    printArray(arr, 6, sizeof(int), printInt);
    printf("]\n");

    printf("  String array: [");
    printArray(strarr, 5, sizeof(char*), printString);
    printf("]\n\n");

    printf("[After Sorting]\n");
    if (order == ASCENDING || order == DESCENDING)
    {
        bubbleSort(arr, 6, sizeof(int), order, intCompare);
        printf("  Integer array: [");
        printArray(arr, 6, sizeof(int), printInt);
        printf("]\n");

        bubbleSort(strarr, 5, sizeof(char*), order, stringCompare);
        printf("  String array: [");
        printArray(strarr, 5, sizeof(char*), printString);
        printf("]\n");
    }
    else
    {
        printf("  Unknown sorting command!\n");
```

```
    }

    return 0;
}
```

待同學實作完畢後，請將所有函式的實作放入以下檔案並繳交：

- **sort.c**：含有上述函式的實作
- **sort.h**：含有上述函式的宣告與定義

本題將以下面的 Makefile 進行批改：

```
all: main.c sort.o
    gcc main.c sort.o
sort.o: sort.c sort.h
    gcc -c sort.c
clean:
    rm -rf *.*~ *~ a.* *.o *.o~
```

本題範例輸出如下：

```
[3:23▲user@ws▲p2]▲./a.out↵
Enter▲sort▲direction▲(1:▲Ascending,▲-1:▲Descending):▲-1↵
[Before▲Sorting]↵
▲▲Integer▲array:▲[5,▲2,▲9,▲1,▲5,▲6]↵
▲▲String▲array:▲[Banana,▲apple,▲Cherry,▲date,▲Egg]↵
↵
[After▲Sorting]↵
▲▲Integer▲array:▲[9,▲6,▲5,▲5,▲2,▲1]↵
▲▲String▲array:▲[date,▲apple,▲Egg,▲Cherry,▲Banana]↵
[3:23▲user@ws▲p2]▲./a.out↵
Enter▲sort▲direction▲(1:▲Ascending,▲-1:▲Descending):▲1↵
[Before▲Sorting]↵
▲▲Integer▲array:▲[5,▲2,▲9,▲1,▲5,▲6]↵
▲▲String▲array:▲[Banana,▲apple,▲Cherry,▲date,▲Egg]↵
↵
[After▲Sorting]↵
▲▲Integer▲array:▲[1,▲2,▲5,▲5,▲6,▲9]↵
▲▲String▲array:▲[Banana,▲Cherry,▲Egg,▲apple,▲date]↵
[3:23▲user@ws▲p2]
```

- 本題相關的程式碼路徑已註明於檔名右側，同學們可以透過路徑複製到自己的家目錄。
- 本題將以上述提供的 Makefile 進行編譯，請同學作答時 **“ 以題目提供的文件為主 ”**。
- 大寫字母的 ASCII 值小於小寫字母，因此在字串遞增排序中，`Banana` 排在 `apple`

前面為正確且預期的結果。
- 本題應繳交檔案如下（**main.c** 及 **Makefile** 則不需繳交）：
  - **sort.c**
  - **sort.h**

---

p3 五子棋-第二次預賽

經歷第一次的預賽，相信同學已經知道自己五子棋的棋力，請同學持續修改並增進自己的五子棋程式。

同學修改完後，請將修改完的五子棋重新上傳至平台：國立屏東大學資工系電腦五子棋 AI 競賽平台上(同樣使用gomoku2026為turnin code)預計將會於2026/06/04再次舉行預賽，所有參賽同學的作品將進行兩兩對奕比試10場，其中5場先手，5場後手，勝負記點採用:勝者2分、平手1分、敗者0分（每局獲勝規則為:黑子先行，五子連線為勝利）。預賽完成後依最終每位參賽同學所得到的分數給定此次作業配分：

- 前 **8%**：本題分數 **$\times 8$**
- 前 **20%**：本題分數 **$\times 4$**
- 前 **50%**：本題分數 **$\times 2$**
- 前 **95%**：本題分數 **$\times 1$**
- **未上傳者** 或 **編譯錯誤** 將以 **0分** 計算

- 本題請使用**gomoku2026作為turnin code上傳至參賽平台** 並 **檢查是否能夠由平台順利完成編譯** 即完成本題目，**請不要繳交至c.hw8**

- 請勿過度使用**AI撰寫**，若是助教發現疑似使用AI撰寫，將會約談同學。
- 約談過程中，若對本人繳交之代碼內容回答模糊、邏輯不通，將會取消同學分數，並以0分計。