Turnin作業7

- Turnin Code: cpp.hw7
- Due Date: 4/20 Sunday 23:59 (midnight) **Hard Deadline**
- 本次作業繳交期限為4月20日週日晚上11點59分0秒！

繳交方式說明

本次Turnin作業包含多個程式題，建議同學可以為這次作業先建立一個資料夾hw7，然後在該資料夾內再為每一題建立一個子資料夾，用以進行每一題的作答以及上傳。每一題的子資料夾名稱已寫於題目前方，請務必依照題目的規定建立子資料夾，例如第1題為p1，第2題為p2，餘依此類推。當我們完成某一個題目的作答後，就可以使用turnin指令將該題的答案上傳。以第1題為例，當我們在p1子資料夾裡完成作答後，就可以回到hw7資料夾，使用以下指令將其上傳：

[user@ws hw7]$ turnin␣cpp.hw7␣p1↵

當然，你也可以等到所有題目都完成後，再回到hw7資料夾，使用以下指令將所有題目都加以上傳：

[user@ws hw7]$ turnin␣cpp.hw7␣.↵



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！

當你完成此次作業的繳交後，可以使用turnin指令的-ls參數，查看已繳交的結果。若已經正確地依要求繳交此次作業，那麼你將可以看到類似以下的查詢結果：

```
[user@ws cpp.hw7]$ turnin -ls cpp.hw7
.:
total 20
drwxrwx---. 2 turninman turnin 4096 Apr  8 10:33 p1
drwxrwx---. 2 turninman turnin 4096 Apr  8 10:33 p2
drwxrwx---. 2 turninman turnin 4096 Apr  8 10:33 p3
drwxrwx---. 2 turninman turnin 4096 Apr  8 10:33 p4
drwxrwx---. 2 turninman turnin 4096 Apr  8 10:33 p5

./p1:
total 4
-rw-rw----. 1 turninman turnin 255 Apr  8 10:33 ball.cpp

./p2:
total 4
-rw-rw----. 1 turninman turnin 3333 Apr  8 10:33 contact.cpp
```

```
./p3:
total 4
-rw-rw----. 1 turninman turnin 264 Apr  8 10:33 dynArray.cpp

./p4:
total 4
-rw-rw----. 1 turninman turnin 791 Apr  8 10:33 nums.cpp

./p5:
total 4
-rw-rw----. 1 turninman turnin 618 Apr  8 10:33 box.cpp
```

【注意：以上的執行結果僅供參考，包含檔案上傳的日期、時間、大小等皆會依實際情況有所不同，請自行仔細檢查是否有正確繳交。】

若是發現自己繳交錯誤的同學，也可以使用以下的指令，將此次作業所有已上傳的檔案與資料夾全部清空：

[user@ws hw7]$ `turnin▲-rm▲cpp.hw7▲.↵`



可以在寫作業前先輸入以下指令獲取這個功課的資料夾 :)

```
cp -r /home/stu/public/cpp2025s/cpp.hw7 .
```

功課資料夾的檔案樹如下：

```
[user@ws cpp.hw7]$ tree .
.
├── p1
│   ├── Makefile
│   ├── ball.h
│   └── main.cpp
├── p2
│   ├── Makefile
│   ├── contact.h
│   ├── in.1
│   ├── in.2
│   └── main.cpp
├── p3
│   ├── Makefile
│   ├── dynArray.h
│   └── main.cpp
├── p4
│   ├── Makefile
│   ├── main.cpp
│   └── nums.h
└── p5
```

```
    ├── Makefile
    ├── box.h
    └── main.cpp

6 directories, 17 files
```

可以在**繳交功課**前先使用tree指令確認資料夾下的檔案~

```
[user@ws cpp.hw7]$ tree .
.
├── p1
│   └── ball.cpp
├── p2
│   └── contact.cpp
├── p3
│   └── dynArray.cpp
├── p4
│   └── nums.cpp
└── p5
    └── box.cpp

6 directories, 5 files
```



同學們如果要獲取題目中的標頭檔或是cpp檔案，可以到/home/stu/public/cpp2025s/cpp.hw7/的各個子資料夾獲取題目的檔案歐！(雖然下面有指令可以直接輸入但還是提醒一下)

p1 比較球的大小



如果尚未獲取題目資料夾可以在p1資料夾使用下列指令：

```
cp -r /home/stu/public/cpp2025s/cpp.hw7/p1/* .
```

請參考以下的程式碼 main.cpp 與 ball.h
filename: main.cpp

```
#include <iostream>
#include <string>
using namespace std;
```

```
#include "ball.h"

int main()
{
    ball b1 = {200, "Ball#1"};
    ball b2 = {.label = "Ball#2"};
    ball *max;
    cout << "Please input the value of Ball#2: ";
    cin >> b2.value;
    max = maxBall(&b1, &b2);
    cout << "The ball with the larger value is ";
    showABall(*max);
}
```

filename: ball.h

```
struct ball
{
    int value;
    char label[10];
};

void showABall(ball b);
ball *maxBall(ball *b1, ball *b2);
```

你必須完成名為 ball.cpp 的cpp語言程式，其中包含定義在 ball.h 的 showABall() 以及 maxBall() 函式的實作，其中 showBall() 函式將印出一個 ball 結構體的 value 及 label 內容; maxBall() 函式則會將所傳入的兩個ball結構體的指標所指向的結構體進行其 value 的比較，並將比較大者的記憶體位址傳回。此題可使用以下的 Makefile 進行編譯：

filename: Makefile

```
all: main.cpp ball.o
    c++ main.cpp ball.o

ball.o: ball.cpp ball.h
    c++ -c ball.cpp

clean:
    rm -f *.o *~ *.*~ a.out
```

此題的執行結果可參考如下：

```
[user@ws hw] ./a.out↵
Please▲input▲the▲value▲of▲Ball#2:▲300↵
The▲ball▲with▲the▲larger▲value▲is▲Ball#2(value=300)↵
[user@ws hw] ./a.out↵
Please▲input▲the▲value▲of▲Ball#2:▲20↵
The▲ball▲with▲the▲larger▲value▲is▲Ball#1(value=200)↵
[user@ws hw] ./a.out↵
Please▲input▲the▲value▲of▲Ball#2:▲100↵
```

```
The▲ball▲with▲the▲larger▲value▲is▲Ball#1(value=200)↵
[user@ws hw]
```



由於該題並未根據相同球的大小來進行比較，所以該題並沒有“ 當兩球形狀同大小 ”的測試檔，請同學放心設計程式! :)

p2 整理聯絡人資料



如果尚未獲取題目資料夾可以在p2資料夾使用下列指令：

```
cp -r /home/stu/public/cpp2025s/cpp.hw7/p2/* .
```

請參考以下的程式碼 main.cpp 與 contact.h
filename : 'main.cpp'

```cpp
#include <iostream>
#include <string>
using namespace std;
#include "contact.h"

int main()
{
    int i;

    Contact mycontacts[numContact];
    for (i = 0; i < numContact; i++)
        mycontacts[i] = getAContact();

    cout << "Sorted by name..." << endl;
    sortContacts(mycontacts, name);
    showAllContacts(mycontacts);

    cout << "Sorted by age..." << endl;
    sortContacts(mycontacts, age);
    showAllContacts(mycontacts);
}
```

filename : contact.h

```
#define numContact 3

enum Gender {Male, Female};
enum Month {January=0, February, March, April, May, June, July, August,
September, October, November, December};
struct Date
{
   Month month;
   short day;
   short year;
};
struct Name
{
   char firstname[20];
   char lastname[10];
};
struct Landline
{
   char areacode[4];
   char number[9];
};
struct Contact
{
   Name name;
   Gender gender;
   Date birthday;
   Landline phone;
};

enum Criteria {name, age} ;

Contact getAContact();
void showAContact(Contact c);
void showAllContacts(Contact cs[]);
void sortContacts(Contact cs[], Criteria c);
```

設計一個CPP語言程式 contact.cpp ，其中包含定義在 contact.h 裡的 showAContact()、getAContact()、showAllContacts()以及 sortContacts() 函式的實作。此題可使用以下的 Makefile 進行編譯。

filename : 'Makefile'

```
all: main.cpp contact.o
    c++ main.cpp contact.o

contact.o: contact.cpp contact.h
    c++ -c contact.cpp

clean:
    rm -f *.o *~ *.*~ a.out
```

此程式完成後將可以接收使用者所輸入的三個聯絡人的資料，並呼叫 sortContacts() 函式進行排序後輸出。請注意呼叫 sortContacts() 函式所傳入的第二個參數是定義為 Criteria 的列舉型態（其值可為 name 或 age），依其值分成兩種處理方式：

1. 當其為 name 時，請依照聯絡人的姓名進行排序—先依據 lastname（姓）由小到大排序，當 lastname 相同時再依 firstname（名字）由小到大排序（本題所使用的測試將不會包含兩個聯絡人同名同姓的情況）。此處所為的由小到大係指依字典中的順序加以輸出，又稱為 Lexicographical Order（例如 Apple 小於 Banana）
2. 當其為 age 時，請依照聯絡人的年齡（以其生日為依據）由大到小進行排序（本題所使用的測試檔將不會包含兩個聯絡人同年同月同日生的情況）。

此題的參考結果可參考如下：

```
[user@ws hw] ./a.out↵
Jun▲Wu▲M▲1972/2/28▲(08)1234567↵
Joe▲Javey▲F▲1973/06/21▲(02)22118888↵
Bruce▲Wu▲M▲1973/4/1▲(02)22117777↵
Sorted▲by▲name...↵
▲▲▲Joe▲Javey▲(Female),▲June▲21st,▲1973,▲(02)22118888.↵
▲▲▲Bruce▲Wu▲(Male),▲April▲1st,▲1973,▲(02)22117777.↵
▲▲▲Jun▲Wu▲(Male),▲February▲28th,▲1972,▲(08)1234567.↵
Sorted▲by▲age...↵
▲▲▲Jun▲Wu▲(Male),▲February▲28th,▲1972,▲(08)1234567.↵
▲▲▲Bruce▲Wu▲(Male),▲April▲1st,▲1973,▲(02)22117777.↵
▲▲▲Joe▲Javey▲(Female),▲June▲21st,▲1973,▲(02)22118888.↵
[user@ws hw] ./a.out↵
Joe▲Chang▲M▲2000/1/2▲(08)8888888↵
Alex▲Cheung▲M▲2000/1/1▲(03)3333333↵
Bony▲Benity▲F▲2000/1/3▲(02)22116666↵
Sorted▲by▲name...↵
▲▲▲Bony▲Benity▲(Female),▲January▲3rd,▲2000,▲(02)22116666.↵
▲▲▲Joe▲Chang▲(Male),▲January▲2nd,▲2000,▲(08)8888888.↵
▲▲▲Alex▲Cheung▲(Male),▲January▲1st,▲2000,▲(03)3333333.↵
Sorted▲by▲age...↵
▲▲▲Alex▲Cheung▲(Male),▲January▲1st,▲2000,▲(03)3333333.↵
▲▲▲Joe▲Chang▲(Male),▲January▲2nd,▲2000,▲(08)8888888.↵
▲▲▲Bony▲Benity▲(Female),▲January▲3rd,▲2000,▲(02)22116666.↵
[user@ws hw]
```



同學們好，考慮到輸入通訊錄資訊可能需要耗費比較多時間，所以這一題有提供 in.1 和 in.2 的文字檔給同學們進行測試，已經儲存在題目資料夾中，指令如下：

```
./a.out < in.1
```

該指令為輸入重定向（Input Redirection）可以將文字檔的內容當作是輸入提供給執行中的程

式。

p3 動態配置記憶體

如果尚未獲取題目資料夾可以在p3資料夾使用下列指令：

```
cp -r /home/stu/public/cpp2025s/cpp.hw7/p3/* .
```

請參考以下的程式碼 main.cpp 與 dynArray.h
filename: main.cpp

```
#include <iostream>
using namespace std;
#include "dynArray.h"

int main()
{
   int **data;

   data=make2DArray();

   for(int i=0;i<4;i++)
   {
      for(int j=0;j<3;j++)
         cout << data[i][j] << " " ;
      cout << endl;
   }

   for(int i=0;i<4; i++)
      delete [] data[i];
   delete [] data;
   return 0;
}
```

filename: dynArray.h

```
int **make2DArray();
```

你必須完成名為 dynArray.cpp 的 cpp 程式，其中包含定義在 dynArray.h 裡的 make2DArray() 函式的實作，此函式會動態配置一個4×3的二維 int 整數陣列，並將其數值設定成以下陣列。

1	**2**	**3**
4	5	6
7	8	9

1	2	3
10	11	12

此題可使用以下的 Makefile 進行編譯。
filename: Makefile

```
all: main.cpp dynArray.o
    c++ main.cpp dynArray.o

dynArray.o: dynArray.cpp dynArray.h
    c++ -c dynArray.cpp

clean:
    rm -f *.o *~ *.*~ a.out
```

```
[user@ws hw] ./a.out↵
1▲2▲3↵
4▲5▲6↵
7▲8▲9↵
10▲11▲12↵
[user@ws hw]
```

p4 動態調整陣列大小



如果尚未獲取題目資料夾可以在p4資料夾使用下列指令：

```
cp -r /home/stu/public/cpp2025s/cpp.hw7/p4/* .
```

請參考以下的程式碼 main.cpp 與 nums.h
filename: main.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "nums.h"
using namespace std;


int main()
{
   int *nums;
```

```
    bool quit=false;
    char cmd;
    int size=5;
    int i,j=1;

    nums= new int[size];
    for(i=0;i<size;i++)
    {
       nums[i]=j++;
       j=j==4?1:j;
    }

    while(!quit)
    {
       cmd=getchar();
       switch(cmd)
       {
          case 'l':
             showAllNumbers(nums, size);
             getchar();
             break;
          case 'i':
             nums=increasingSpace(nums, size);
             size*=2;
             getchar();
             break;
          case 'd':
             nums=decreasingSpace(nums, size);
             size=size<=5?size:size/=2;
             getchar();
             break;
          case 'q':
             quit=true;
             break;
       }
    }
}
```

filename: nums.h

```
void showAllNumbers(int nums[], int size);
int *increasingSpace(int nums[], int size);
int *decreasingSpace(int nums[], int size);
```

請設計一個CPP語言的程式nums.cpp，其中包含定義在 nums.h 裡的 showAllNumbers()，increasingSpace() 與 decreasingSpace() 函式的實作。此程式預先建立了一個大小為5的動態陣列空間nums，然後在執行時依使用者指令，進行以下操作：

- l:呼叫showAllNumbers() 列示 nums 中所有的數字。
- i:呼叫increasingSpace() 增加一倍空間。
- d:呼叫decreasingSpace() 減少一半空間，但nums的大小不得小於5。

要注意的是，不論 nums 中有多少空間，其所存放的皆為數字1、2與3，並且會維持此依序（請自行觀察本的執行結果）。
此題可使用以下的Makefile 進行編譯。
filename: Makefile

```
all: main.cpp nums.o
    c++ main.cpp nums.o

nums.o: nums.cpp nums.h
    c++ -c nums.cpp

clean:
    rm -f *.o *~ *.*~ a.out
```

此題執行結果可參考如下：

```
[user@ws hw] ./a.out↵
l↵
12312↵
q↵
[user@ws hw] ./a.out↵
i↵
i↵
l↵
12312312312312312312↵
d↵
l↵
1231231231↵
d↵
l↵
12312↵
d↵
Cannot▲resize▲it!↵
l↵
12312↵
q↵
[user@ws hw]
```

p5 動態陣列



如果尚未獲取題目資料夾可以在p5資料夾使用下列指令：

```
cp -r /home/stu/public/cpp2025s/cpp.hw7/p5/* .
```

請參考以下的程式碼 main.cpp 與 box.h
filename :main.cpp

```
#include<iostream>
using namespace std;
#include "box.h"

int main()
{
    int *box;
    box=makeBox();
    showBox(box);
    descendingBox(box);
    showBox(box);
    delete [] box;
}
```

filename:box.h

```
void showBox(int *box);
void descendingBox(int *box);
int *makeBox();
```

請完成一個 cpp 程式名為 box.cpp，其中包含定義在 box.h 裡的 showBox()，decendingBox() 與 makeBox() 函式的實作，其中 makeBox() 函式將會動態配置一個可存放10個 int 整數的陣列空間，並將整數值1~10依序放入該陣列中再將其記憶體位址回傳; decendingBox() 則將所傳入的記憶體位址視為一個 int(10) 整數陣列，並使用降序排列的方式轉換整個陣列的排序（由大到小 p.s. sort可以複習下）。至於 showBox() 同樣將所傳入的記憶體視為一個 int[10] 整數陣列，並將其內容輸出。此題可使用以下的 Makefile 進行編譯。
filename: Makefile

```
all: main.cpp box.o
    c++ main.cpp box.o

box.o: box.cpp box.h
    c++ -c box.cpp

clean:
    rm -f *.o *~ *.*~ a.out
```

此題的執行結果可參考如下。

```
[user@ws hw] ./a.out↵
1,2,3,4,5,6,7,8,9,10↵
10,9,8,7,6,5,4,3,2,1↵
[user@ws hw]
```

From:
https://junwu.nptu.edu.tw/dokuwiki/ - **Jun Wu的教學網頁**
國立屏東大學資訊工程學系
CSIE, NPTU
Total: 339282

Permanent link:
https://junwu.nptu.edu.tw/dokuwiki/doku.php?id=cpp:2025spring:hw7

Last update: **2025/04/22 01:54**