**C++ 程式語言期末專題挑戰**

組員:

00781003李昌樺

00781035曾鈺皓

00781023杜　全

00781019黃育騏

00781033鄭鈞元

00781047黃旭毅

1.程式碼

This program includes the following files :

makefile

main.cpp

tools.cpp

tools.h

InputFunction.cpp

InputFunction.h

CalculateFunction.cpp

CalculateFunction.h

debug.cpp

debug.h

makefile

main : main.cpp tools.cpp InputFunction.cpp CalculateFunction.cpp

g++ main.cpp tools.cpp debug.cpp InputFunction.cpp CalculateFunction.cpp -o Final.exe

main.cpp

#include"tools.h"

#include"InputFunction.h"

#include"CalculateFunction.h"

//#define debug

#include"debug.h"

/\* Final Project(ANNSFM)

Structure:

Data:

M = 8 N = 12 L = 1n = 708

ANNSFM\_Config\_I:M \* 2組

ANNSFM\_Config\_O:L \* 2組

ANNSFM\_Config\_S: 3個

ANNSFM\_CS\_HB: N個

ANNSFM\_CS\_HW: M \* N

ANNSFM\_CS\_OB: L個

ANNSFM\_CS\_OW: N \* L

ANNSFM\_data\_size:1個

ANNSFM\_inputs: M \* n組

Function:

ANNSFM\_Config\_S每個值分別對應Input Layer, Hidden Layer, Outper Layer之數量,必須優先處理

ANNSFM\_Config\_I為先使用者,之後是ANNSFM\_CS\_HB,ANNSFM\_CS\_HW

ANNSFM\_Config\_O為輸出時使用,之後是ANNSFM\_CS\_OB,ANNSFM\_CS\_OW

ANNSFM\_data\_size是處理ANNSFM\_inputs使用,最後再進行

Process:

Inputs

1.Input ANNSFM\_Config\_S

2.Input ANNSFM\_Config\_I

3.Input ANNSFM\_CS\_HB

4.Input ANNSFM\_CS\_HW

5.Input ANNSFM\_Config\_O

6.Input ANNSFM\_CS\_OB

7.Input ANNSFM\_CS\_OW

8.Input ANNSFM\_data\_size

9.Input ANNSFM\_inputs

Calculate

1.Input Layer Normalization -> Into Hidden layer

2.Hidden Layer Summation WH(n,m)\*I(m,l) + BH(n,1)

3.Hidden Layer Transfer Function -> Into Output layer

4.Output Layer Summation WO(1,n)\*I(m,1) + BO(l,1)

5.Output Layer Inverse Normalization

Output

1.Output y into ANNSFM\_outputs

Graph

1.Graph with Excel or Matlab

\*/

int main(void){

#ifdef debug

fstream F;

#endif

string buffer;

//Input ANNSFM\_Config\_S

fstream FILE\_Config\_S;

File\_OpenforRead(FILE\_Config\_S,"Data\\ANNSFM\_Config\_S.txt");

Config\_S\_Struct \*Config\_S = new Config\_S\_Struct;

Input\_Config\_S(FILE\_Config\_S,Config\_S);

#ifdef debug

Debug\_debug\_Config\_S(F,"debugs\\debug\_Config\_S.txt",Config\_S);

#endif

File\_Close(FILE\_Config\_S,"Data\\ANNSFM\_Config\_S.txt");

//Input ANNSFM\_Config\_I

fstream FILE\_Config\_I;

File\_OpenforRead(FILE\_Config\_I,"Data\\ANNSFM\_Config\_I.txt");

double \*Config\_I\_Min = Input\_Config\_IO\_MaxMin(FILE\_Config\_I,Config\_S->Input\_Layer\_Count);

double \*Config\_I\_Max = Input\_Config\_IO\_MaxMin(FILE\_Config\_I,Config\_S->Input\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_IO\_MinMax(F,"debugs\\debug\_Config\_I.txt",Config\_S->Input\_Layer\_Count,Config\_I\_Min,Config\_I\_Max);

#endif

File\_Close(FILE\_Config\_I,"Data\\ANNSFM\_Config\_I.txt");

//Input ANNSFM\_CS\_HB

fstream FILE\_CS\_HB;

File\_OpenforRead(FILE\_CS\_HB,"Data\\ANNSFM\_CS\_HB.txt");

double \*Config\_HB = Input\_CS\_HB\_and\_OB(FILE\_CS\_HB,Config\_S->Hidden\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_HB\_and\_OB(F,"debugs\\debug\_Config\_HB.txt",Config\_S->Hidden\_Layer\_Count,Config\_HB);

#endif

File\_Close(FILE\_CS\_HB,"Data\\ANNSFM\_CS\_HB.txt");

//Input ANNSFM\_CS\_HW

fstream FILE\_CS\_HW;

File\_OpenforRead(FILE\_CS\_HW,"Data\\ANNSFM\_CS\_HW.txt");

Matrix \*CS\_HW\_head = Input\_CS\_HW\_and\_OW(FILE\_CS\_HW,Config\_S->Hidden\_Layer\_Count,Config\_S->Input\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_HW\_and\_OW(F,"debugs\\debug\_Config\_HW.txt",Config\_S->Hidden\_Layer\_Count,Config\_S->Input\_Layer\_Count,CS\_HW\_head);

#endif

File\_Close(FILE\_CS\_HW,"Data\\ANNSFM\_CS\_HW.txt");

//Input ANNSFM\_Config\_O

fstream FILE\_Config\_O;

File\_OpenforRead(FILE\_Config\_O,"Data\\ANNSFM\_Config\_O.txt");

double \*Config\_O\_Min = Input\_Config\_IO\_MaxMin(FILE\_Config\_O,Config\_S->Output\_Layer\_Count);

double \*Config\_O\_Max = Input\_Config\_IO\_MaxMin(FILE\_Config\_O,Config\_S->Output\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_IO\_MinMax(F,"debugs\\debug\_Config\_O.txt",Config\_S->Output\_Layer\_Count,Config\_O\_Min,Config\_O\_Max);

#endif

File\_Close(FILE\_Config\_O,"Data\\ANNSFM\_Config\_O.txt");

//Input ANNSFM\_CS\_OB

fstream FILE\_CS\_OB;

File\_OpenforRead(FILE\_CS\_OB,"Data\\ANNSFM\_CS\_OB.txt");

double \*Config\_OB = Input\_CS\_HB\_and\_OB(FILE\_CS\_OB,Config\_S->Output\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_HB\_and\_OB(F,"debugs\\debug\_Config\_OB.txt",Config\_S->Output\_Layer\_Count,Config\_OB);

#endif

File\_Close(FILE\_CS\_OB,"Data\\ANNSFM\_CS\_OB.txt");

//Input ANNSFM\_CS\_OW

fstream FILE\_CS\_OW;

File\_OpenforRead(FILE\_CS\_OW,"Data\\ANNSFM\_CS\_OW.txt");

Matrix \*CS\_OW\_head = Input\_CS\_HW\_and\_OW(FILE\_CS\_OW,Config\_S->Output\_Layer\_Count,Config\_S->Hidden\_Layer\_Count);

#ifdef debug

Debug\_debug\_Config\_HW\_and\_OW(F,"debugs\\debug\_Config\_OW.txt",Config\_S->Output\_Layer\_Count,Config\_S->Hidden\_Layer\_Count,CS\_OW\_head);

#endif

File\_Close(FILE\_CS\_OW,"Data\\ANNSFM\_CS\_OW.txt");

//Input ANNSFM\_data\_size

fstream FILE\_data\_size;

int \*Data\_Size = new int;

File\_OpenforRead(FILE\_data\_size,"Data\\ANNSFM\_data\_size.txt");

FILE\_data\_size >> buffer;

\*Data\_Size = stoi(buffer);

File\_Close(FILE\_data\_size,"Data\\ANNSFM\_data\_size.txt");

//Input ANNSFM\_inputs

fstream FILE\_inputs;

File\_OpenforRead(FILE\_inputs,"Data\\ANNSFM\_inputs.txt");

Matrix \*inputs\_head = Input\_inputs(FILE\_inputs,Data\_Size,Config\_S->Input\_Layer\_Count);

#ifdef debug

Debug\_debug\_inputs(F,"debugs\\debug\_inputs.txt",inputs\_head,Config\_S);

#endif

File\_Close(FILE\_inputs,"Data\\ANNSFM\_inputs.txt");

fstream FILE\_outputs;

File\_OpenforWrite(FILE\_outputs,"Data\\ANNSFM\_outputs.txt");

Matrix \*CurrentData = inputs\_head;

double \*Hn = new double[Config\_S->Hidden\_Layer\_Count];

double \*Ol =new double[Config\_S->Output\_Layer\_Count];

for(int i = 0 ; i < \*Data\_Size; i++){

cout << "Cal Data " << i <<endl;

//Input Layer Normalization -> Into Hidden layer

Input\_Layer\_Normalization(CurrentData->data,Config\_I\_Min,Config\_I\_Max,Config\_S);

#ifdef debug

Debug\_Input\_Layer\_Normalization(F,"debugs\\debug\_Input\_Layer\_Normalization.txt",CurrentData,Config\_S);

#endif

//Hidden Layer Summation WH(n,m)\*I(m,l) + BH(n,1)

//Hidden Layer Transfer Function -> Into Output layer

Hidden\_Layer\_Summation(CurrentData->data,Hn,CS\_HW\_head,Config\_HB,Config\_S);

Hidden\_Layer\_Transfer(Hn,Config\_S);

#ifdef debug

Debug\_Hidden\_Layer\_Transfer(F,"debugs\\debug\_Hidden\_Layer\_Transfer.txt",Hn,Config\_S);

#endif

//Output Layer Summation WO(1,n)\*I(m,1) + BO(l,1)

//Output Layer Inverse Normalization

Output\_Layer\_Summation(Hn,Ol,CS\_OW\_head,Config\_OB,Config\_S);

Output\_Layer\_Inverse\_Normalization(Ol,Config\_O\_Min,Config\_O\_Max,Config\_S);

//Output y into ANNSFM\_outputs

Output\_ANNSFM\_outputs(FILE\_outputs,Ol,Config\_S);

CurrentData = CurrentData->Next;

}

delete Hn;

delete Ol;

File\_Close(FILE\_outputs,"Data\\ANNSFM\_outputs.txt");

return 0;

}

tools.cpp

#include"tools.h"

void File\_OpenforRead(fstream& F,string S){

F.open(S.c\_str(),ios\_base::in);

if(F.fail()){

cout << "Open Error: " << S << endl;

system("pause");

}

}

void File\_OpenforWrite(fstream& F,string S){

F.open(S.c\_str(),ios\_base::out);

if(F.fail()){

cout << "Open Error: " << S << endl;

system("pause");

}

}

void File\_OpenforAppend(fstream& F,string S){

F.open(S.c\_str(),ios\_base::app);

if(F.fail()){

cout << "Open Error: " << S << endl;

system("pause");

}

}

void File\_Close(fstream& F,string S){

F.clear();

F.close();

if(F.fail()){

cout << "Close Error: " << S << endl;

system("pause");

}

}

matrix \*Create\_Matrix(fstream& F,int I,int J){

Matrix \*inputs\_head = new Matrix;

Matrix \*inputs\_current = inputs\_head;

for(int a = 0; a < I; a++){

double \*array\_buffer = new double[J] ;

inputs\_current->data = array\_buffer;

Input\_nData\_Into\_Array(F,J,array\_buffer);

if(!(a == I - 1)){

//new node

inputs\_current->Next = new Matrix;

inputs\_current = inputs\_current->Next;

}

}

return inputs\_head;

}

void Input\_nData\_Into\_Array(fstream& F,int Layer\_Count,double \*array\_buffer){

string buffer;

for(int i = 0; i < Layer\_Count; i++){

F >> buffer;

\*(array\_buffer+i) = stod(buffer);

}

}

tools.h

#include<iostream>

#include<cstdlib>

#include<fstream>

#include<string>

#include<cmath>

using namespace std;

#ifndef StructDefininition

#define StructDefininition

typedef struct config\_s\_struct{

int Input\_Layer\_Count = 0;

int Hidden\_Layer\_Count = 0;

int Output\_Layer\_Count = 0;

} Config\_S\_Struct;

typedef struct matrix{

double \*data = NULL;

matrix \*Next = NULL;

}Matrix;

#endif

void File\_OpenforRead(fstream& F,string S);

void File\_OpenforWrite(fstream& F,string S);

void File\_OpenforAppend(fstream& F,string S);

void File\_Close(fstream& F,string S);

matrix \*Create\_Matrix(fstream& F,int I,int J);

void Input\_nData\_Into\_Array(fstream& F,int Layer\_Count,double \*array\_buffer);

InputFunction.cpp

#include"InputFunction.h"

void Input\_Config\_S(fstream& F,Config\_S\_Struct \*C){

string buffer;

F >> buffer;

C->Input\_Layer\_Count = stoi(buffer);

F >> buffer;

C->Hidden\_Layer\_Count = stoi(buffer);

F >> buffer;

C->Output\_Layer\_Count = stoi(buffer);

}

double \*Input\_Config\_IO\_MaxMin(fstream& F,int Layer\_Count){

double \*array\_buffer = new double[Layer\_Count];

Input\_nData\_Into\_Array(F,Layer\_Count,array\_buffer);

return array\_buffer;

}

double \*Input\_CS\_HB\_and\_OB(fstream& F,int Layer\_Count){

double \*array\_buffer = new double[Layer\_Count];

Input\_nData\_Into\_Array(F,Layer\_Count,array\_buffer);

return array\_buffer;

}

matrix \*Input\_CS\_HW\_and\_OW(fstream& F,int I,int J){

return Create\_Matrix(F,I,J);

}

matrix \*Input\_inputs(fstream& F,int \*Data\_Size,int Layer\_Count){

return Create\_Matrix(F,\*Data\_Size,Layer\_Count);

}

InputFunction.h

#include<iostream>

#include<cstdlib>

#include<fstream>

#include<string>

#include<cmath>

#include"tools.h"

using namespace std;

void Input\_Config\_S(fstream& F,Config\_S\_Struct \*C);

double \*Input\_Config\_IO\_MaxMin(fstream& F,int Layer\_Count);

double \*Input\_CS\_HB\_and\_OB(fstream& F,int Layer\_Count);

matrix \*Input\_CS\_HW\_and\_OW(fstream& F,int I,int J);

matrix \*Input\_inputs(fstream& F,int \*Data\_Size,int Layer\_Count);

CalculateFunction.cpp

#include"CalculateFunction.h"

void Input\_Layer\_Normalization(double \*data,double \*Min,double \*Max,Config\_S\_Struct \*C){

for(int a = 0 ; a < C->Input\_Layer\_Count; a++)

data[a] = 2 \* (data[a] - Min[a]) / (Max[a] - Min[a]) - 1;

}

void Hidden\_Layer\_Summation(double \*data,double \*Hn,matrix \*HW,double \*HB,Config\_S\_Struct \*C){

//Flush

for(int a = 0 ; a < C->Hidden\_Layer\_Count ; a++){

Hn[a] = 0;

}

for(int a = 0; a < C->Hidden\_Layer\_Count ; a++){

for(int M = 0 ; M < C->Input\_Layer\_Count; M++){

Hn[a] += HW->data[M] \* data[M];

}

Hn[a] += HB[a];

HW = HW->Next;

}

}

void Hidden\_Layer\_Transfer(double \*Hn,Config\_S\_Struct \*C){

for(int a = 0 ; a < C->Hidden\_Layer\_Count ; a++)

Hn[a] = (2 / (1 + exp(-2 \* Hn[a]))) - 1;

}

void Output\_Layer\_Summation(double \*Hn,double \*Ol,matrix \*OW,double \*OB,Config\_S\_Struct \*C){

//Flush

for(int a = 0 ; a < C->Output\_Layer\_Count ; a++){

Ol[a] = 0;

}

for(int a = 0; a < C->Output\_Layer\_Count ; a++){

for(int N = 0 ; N < C->Hidden\_Layer\_Count; N++){

Ol[a] += OW->data[N] \* Hn[N];

}

Ol[a] += OB[a];

OW = OW->Next;

}

}

void Output\_Layer\_Inverse\_Normalization(double \*Ol,double \*Min,double \*Max,Config\_S\_Struct \*C){

for(int a = 0 ; a < C->Output\_Layer\_Count; a++)

Ol[a] = (Ol[a]+1)/2 \* (Max[a] - Min[a]) + Min[a];

}

void Output\_ANNSFM\_outputs(fstream& F,double \*Ol,Config\_S\_Struct \*C){

for(int a = 0 ; a < C->Output\_Layer\_Count ; a++){

F.precision(16);

F << Ol[a] << " ";

}

F << endl;

}

CalculateFunction.h

#include<iostream>

#include<cstdlib>

#include<fstream>

#include<string>

#include<cmath>

#include"tools.h"

using namespace std;

void Input\_Layer\_Normalization(double \*data,double \*Min,double \*Max,Config\_S\_Struct \*C);

void Hidden\_Layer\_Summation(double \*data,double \*Hn,Matrix \*HW,double \*HB,Config\_S\_Struct \*C);

void Hidden\_Layer\_Transfer(double \*Hn,Config\_S\_Struct \*C);

void Output\_Layer\_Summation(double \*Hn,double \*Ol,Matrix \*OW,double \*OB,Config\_S\_Struct \*C);

void Output\_Layer\_Inverse\_Normalization(double \*Ol,double \*Min,double \*Max,Config\_S\_Struct \*C);

void Output\_ANNSFM\_outputs(fstream& F,double \*Ol,Config\_S\_Struct \*C);

debug.cpp

#include"debug.h"

void Debug\_debug\_Config\_S(fstream& F,string S,Config\_S\_Struct \*C){

File\_OpenforWrite(F,"debug\_Config\_S.txt");

F << C->Input\_Layer\_Count << endl;

F << C->Hidden\_Layer\_Count << endl;

F << C->Output\_Layer\_Count << endl;

File\_Close(F,"debug\_Config\_S.txt");

}

void Debug\_debug\_Config\_IO\_MinMax(fstream& F, string S, int Layer\_Count,double \*Min,double \*Max){

File\_OpenforWrite(F,S.c\_str());

for(int a = 0; a < Layer\_Count; a++){

F << Min[a] << " ";

}

F << endl;

for(int a = 0; a < Layer\_Count; a++){

F << Max[a] << " ";

}

File\_Close(F,S.c\_str());

}

void Debug\_debug\_Config\_HB\_and\_OB(fstream& F, string S, int Layer\_Count,double \*ptr){

File\_OpenforWrite(F,S.c\_str());

for(int a = 0; a < Layer\_Count; a++){

F.precision(16);

F << ptr[a] << endl;

}

File\_Close(F,S.c\_str());

}

void Debug\_debug\_Config\_HW\_and\_OW(fstream& F, string S, int I, int J, matrix \*ptr){

File\_OpenforWrite(F,S.c\_str());

for(int a = 0; a < I; a++){

for(int b = 0; b < J; b++){

F.precision(16);

F << ptr->data[b] << " ";

}

F << endl;

ptr = ptr->Next;

}

File\_Close(F,S.c\_str());

}

void Debug\_debug\_inputs(fstream& F, string S, matrix \*ptr, Config\_S\_Struct \*C){

File\_OpenforWrite(F,S.c\_str());

while(!(ptr == NULL)){

for(int a = 0; a < C->Input\_Layer\_Count; a++){

F << ptr->data[a] << " ";

}

F << endl;

ptr = ptr->Next;

}

File\_Close(F,S.c\_str());

}

void Debug\_Input\_Layer\_Normalization(fstream& F, string S, matrix \*ptr, Config\_S\_Struct \*C){

File\_OpenforAppend(F,S.c\_str());

for(int a = 0; a < C->Input\_Layer\_Count; a++){

F << ptr->data[a] << " ";

}

F << endl;

File\_Close(F,S.c\_str());

}

void Debug\_Hidden\_Layer\_Transfer(fstream& F, string S, double \*ptr, Config\_S\_Struct \*C){

File\_OpenforAppend(F,S.c\_str());

for(int a = 0; a < C->Hidden\_Layer\_Count; a++){

F << ptr[a] << " ";

}

F << endl;

File\_Close(F,S.c\_str());

}

debug.h

#include<iostream>

#include<cstdlib>

#include<fstream>

#include<string>

#include<cmath>

#include"tools.h"

using namespace std;

#ifdef debug

void Debug\_debug\_Config\_S(fstream& F,string S,Config\_S\_Struct \*C);

void Debug\_debug\_Config\_IO\_MinMax(fstream& F, string S, int Layer\_Count,double \*Min,double \*Max);

void Debug\_debug\_Config\_HB\_and\_OB(fstream& F, string S, int Layer\_Count,double \*ptr);

void Debug\_debug\_Config\_HW\_and\_OW(fstream& F, string S, int I, int J, matrix \*ptr);

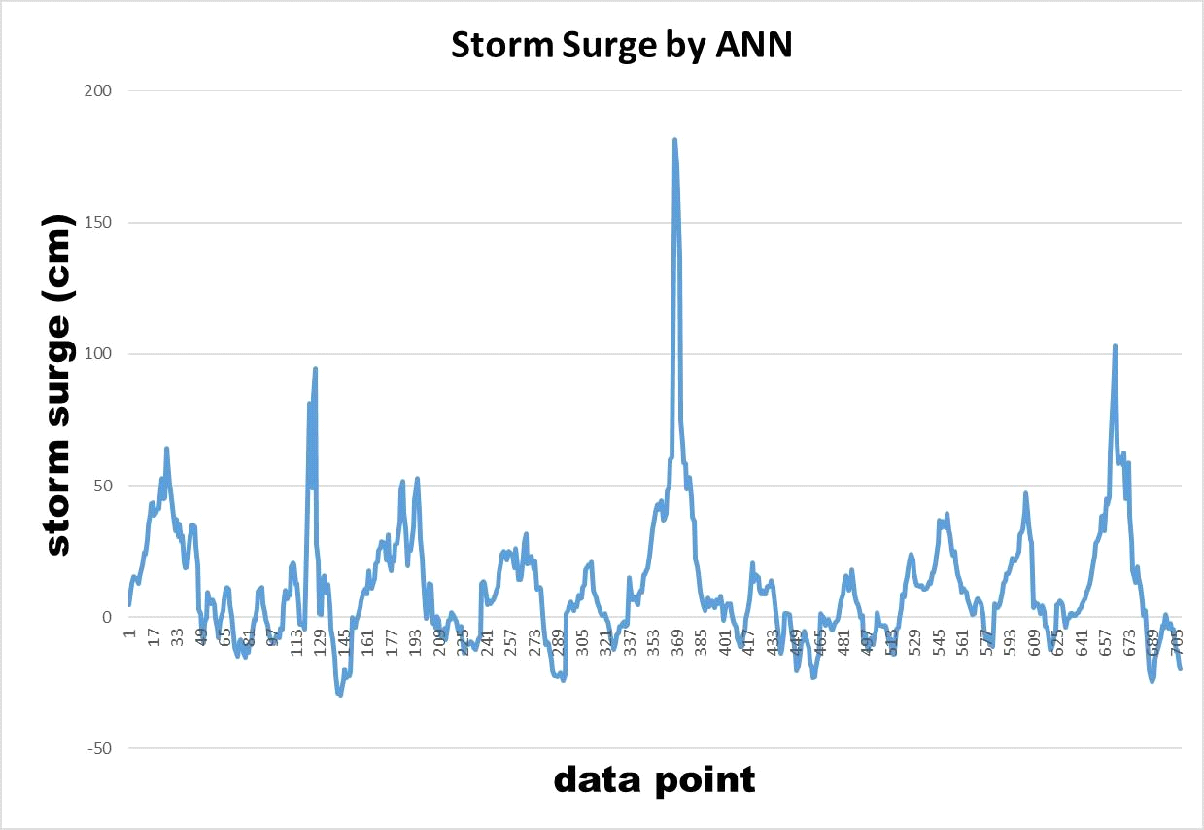
void Debug\_debug\_inputs(fstream& F, string S, matrix \*ptr, Config\_S\_Struct \*C);

void Debug\_Input\_Layer\_Normalization(fstream& F, string S, matrix \*ptr, Config\_S\_Struct \*C);

void Debug\_Hidden\_Layer\_Transfer(fstream& F, string S, double \*ptr, Config\_S\_Struct \*C);

#endif

2.作圖



(3) 學期心得感想與意見以及給自己的評分

00781003李昌樺 :

學期心得

課程由簡入深從一開始的資料型態、函式輸入輸出、程式結構到進入迴圈都是期中考的範圍這部分還算得心應手，進入到了中後期矩陣的加入相較於matlab是非常的複雜也較難使用，剛開始嘗試著以matlab的形式來撰寫程式但卻不斷碰壁，而以迴圈的形式來寫矩陣卻又感覺多此一舉是無法理解並且感到懊惱。期末考的內容大多也跟矩陣有關希望不會在這卡關。C++相較於其他程式語言是相對複雜的沒有精簡的編譯方式需要清晰的邏輯和清楚的排版容錯率相對的小，透過這個學期我才明白程式的深度，還需要跟多的努力和研究才能有將程式應用到生活上的可能，否則單憑我現在的程式水準所寫出來的程式是無法使我自己也滿意的。

感想意見

面對功課和小考都是以開書的形式參考例題上的邏輯來反推完成題目，上課詳細介紹的每一個例題都會獲得某些靈感的出現，在練習時常常試著將自己的想法融入到例題中，像是將課本寫死的部分改成定義的變數利用cin、cout的方式將程式活化應用，或是改編程式的圖形等等...，但到了後半部因為課程時間的關係常常一次練習3-4題且每題內容都偏長時間不太充裕也無法進一步的深入這個程式有點可惜。

自評分數 :90

00781035曾鈺皓 :

**1.心得感想**

C++確實是一個挺麻煩的編譯式程式語言，有許多的細節在撰寫程式時缺一

不可，但學習精神或許就是留意細節，我想這一點對於處理海洋資料這類大範

圍的資訊會有很大的幫助。我認為學習程式語言最大的動機就是能促進大腦思

考，訓練邏輯的過程能夠讓自己的思路更加清晰，並且找到解決問題的最佳方

法。因此，不論在日常生活或是想法上都能對自己有正面的影響。在編寫過程

中雖然有些痛苦，但是每當靠自己寫出一個完整的程式，總是會有強大的成就

感，更重要的是能活化自己的創意，這就是程式語言好玩的地方，不同的人會

寫出不同的演算法，一生之中受用無窮。

常常聽到不同程式語言會有不同的優缺點，在這學期的C++學習過程中，我

感覺到C++在矩陣方面的應用相對於MATLAB來說效率並不是很好，資料處理

上也相對麻煩；再對比Python，可以看到在學習上面對於初學者Python會是相

對C++更好學習與入門的程式語言。而結論是C++或C語言雖然隨著時代演進，

缺點會慢慢浮現，但畢竟C語言還是許多程式語言發展的基礎，也是現代資訊

科學的必修課。

**2.感想意見** :

還是希望能在課堂上有更多的實作時間，也希望日後課程的進行能夠在保留

討論空間的情況下，有個明確的目標與事先在學期初就訂定考試評量方法，避

免評量方式變動太大。

**3.自評分數 : 89**

00781023杜　全 : 1.心得感想

經過了這個學期下來的c++學習，我真的是覺得滿不容易，從遭遇問題到思考如何運用指令解決問題，我覺得是個非常需要專注力的過程；不過這也可以訓練我們處理問題的能力，當遇到問題時該如何處理，該用什麼樣的方式來處理，都是要學習的目標。

2.感想意見

我覺得到了後期的時候，上課速度有一點過快，當還在咀嚼剛接收的新知識時，卻已經來到了下一個主題；以及上課練習的時間有點不夠，有時候篇幅較長需要較多的時間。

3.自評分數：68

00781019黃育騏 :

**學期心得**

　　我覺得這學期對程式語言的學習十分豐富，不管是python還是c++都讓我對新領域的事物有更進一步的了解，雖然學習的時間不長只有短短的一學期，但也讓我了解我的c++有很多要學，以我現在的能力頂多只能解決一些基礎問題，但我想大學應該多多接觸不同方面的知識，即使失敗或是挫折這都是學習的一部分，有了基礎以後如果要自學c#或是其他程式語言也會相對容易，而且我認為學習程式語言最大的好處是邏輯思考，寫程式往往會需要想到前因後果，所以對邏輯思考有很大的益處。

**感想意見**

　　我認為學習是需要練習的，但其實上課可以盡量不要練習課本上的例題，其實把課本例題結合起來練習其實更有效率，因為我都這樣練習，不但減少練習時間，也結合課本想教的技巧，更可以促使大家思考，同學們也可以教學相長。

**自評分數 : 87**

00781033鄭鈞元 :

學期心得

這門課真的令我感觸良多，學會了寫程式的基本理念，雖然我上完課仍然看不懂黃旭毅同學的作業。

感想意見

希望我也有能力能參與期末報告，但是同學太厲害了，我只能在旁邊看。

自評分數 60

00781047黃旭毅 :

**學期心得**

　　做Fianl好爽==

**感想意見**

　　明年開一堂資料結構及演算法教訓一下混分仔好嗎==

**自評分數 :超過100分**

4.組員分工

00781003李昌樺 : 程式分析

00781035曾鈺皓 : 製作圖表

00781023杜　全 : 專案管理

00781019黃育騏 : 製作報告

00781033鄭鈞元 : 製作報告

00781047黃旭毅 : 程式開發