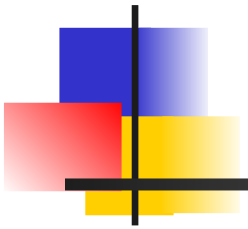
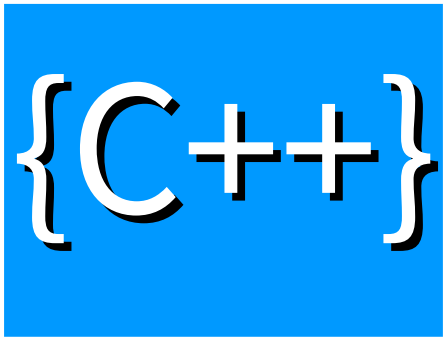
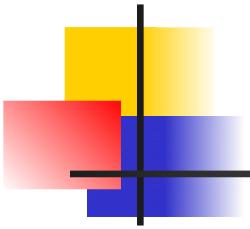


Multiple Source Files

Week 14



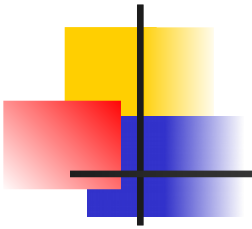
Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw



Multiple Source Files

- Small programs are typically written in a single .cpp file.
- As programs get larger, it's necessary to split the source into several files in order to make the project manageable.

Use .h files to for shared declarations

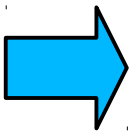


- When using muliple source files, you want to use the same declaration where the function is called as where it's defined

- The solution is to create a header file (.h) which contains the function declarations. By #including this in all source files (.cpp) that use or define this function, consistency is maintained.

Use .h files to for shared declarations

```
#include <cstdio>

void helloWorld();

int main(){
	helloWorld();
}

void helloWorld()
{
	printf("hello world\n");
}
```



main.c

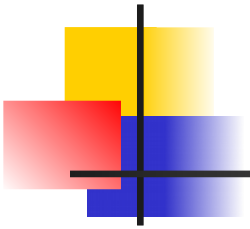
```
#include "helloworld.h"

int main(){
	helloWorld();
}
```

helloworld.h

```
#include <cstdio>

void helloWorld();
```

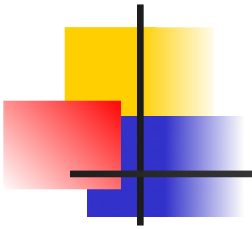
helloworld.c

```
#include "helloworld.h"

void helloWorld()
{
	printf("hello world\n");
}
```



Project

- If you use an IDE (eg, instead of a make file), you often have to create a project and add all source files to it, so that it knows which files belong together.
- A project may also include other, non-source, resource files.



Include guard

- Also known as "macro guard"
- A particular construct used to avoid the problem of double inclusion when dealing with the include directive
- The addition of #include guards to a header file is one way to make that file idempotent.

Double inclusion

- The following C code demonstrates a problem that can arise if #include guards are missing

File "grandfather.h"

```
struct foo {
    int member;
};
```

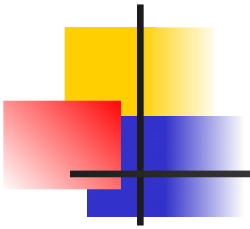
File "father.h"

```
#include "grandfather.h"
```

File "child.c"

```
#include "grandfather.h"
#include "father.h"
```

- The file "child.c" has indirectly included two copies of the text in the header file "grandfather.h".
- This causes a compilation error, since the structure type foo is apparently defined twice.

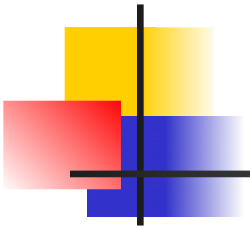


Use of include guards

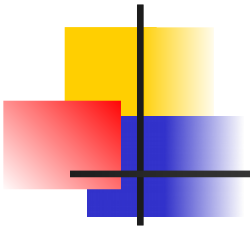
File "grandfather.h"

```
#ifndef GRANDFATHER_H
#define GRANDFATHER_H

struct foo {
    int member;
};

#endif /* GRANDFATHER_H */
```

Example with two source files and a shared header



- Here's a example
  - The main program in one file (averageMain.cpp)
  - A function in another (average.cpp)
  - A header file containing the function prototype (average.h)



Header file - average.h

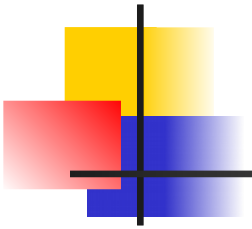
```
// Function prototype.

#ifndef AVERAGE_H
#define AVERAGE_H
float average(const vector<float>& x);
#endif /* AVERAGE_H */
```

Main program - averageMain.cpp

```
// Print above average numbers.  Illustrates multiple source file compilations.
// Standard includes
#include <iostream>
#include <stdlib.h>
#include <vector>
using namespace std;
// Private include for average function.
#include "average.h"

int main(){

    vector<float> fv;    // Store the input floats here.
    float         temp;  // Temp for input.

    //... Read floats into vector
    while (cin >> temp) {
        fv.push_back(temp);
    }

    //... Compute average.
    float avg = average(fv);

    //... Print greater than average numbers.
    cout << "Average = " << avg << endl;
    for (int i = 0; i < fv.size(); i++) {
        if (fv[i] > avg) {
            cout << fv[i] << endl;
        }
    }

    return 0;
}
```

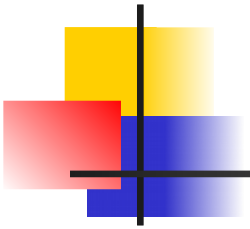
Function definition - average.cpp

```
// Compute average in vector.
// Standard includes
#include <vector>
using namespace std;
// Private header file with prototype for average.
#include "average.h"

// average
float average(const vector<float>& x) {
    float sum = 0.0;
    for (int i=0; i<x.size(); i++) {
        sum += x[i];
    }

    return sum / x.size();
}
```



Assignment

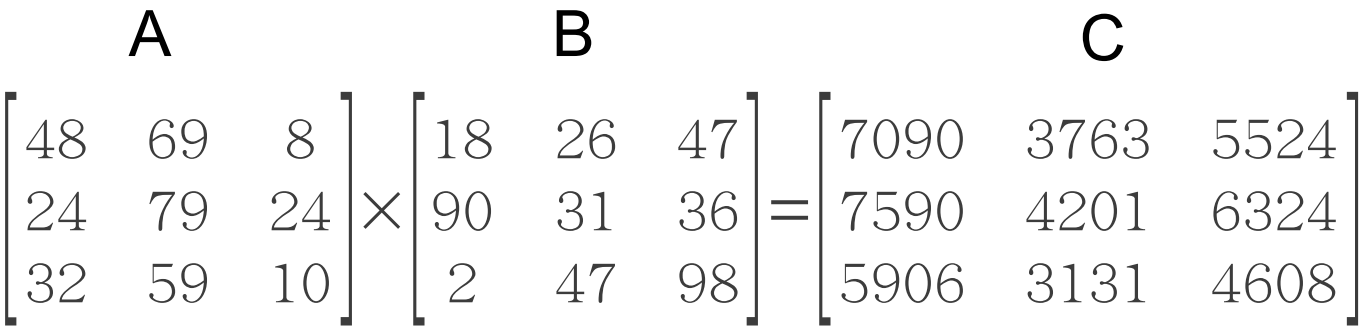
- Write a program to compute the multiplication of two MxN matrix.
- Your program must be satisfied with these conditions
  - Create a project with multiple source files (main.cpp, matrix.cpp, matrix.h)
  - matrix.cpp must contain three functions
    - Load each matrix from a file
    - Compute multiplication of two matrix.You must check if the multiplication is valid or not.
    - Print a matrix



Assignment

- Use three functions you implement in matrix.cpp to complete main.cpp. The main procedure in main.cpp must follow these steps described below
  - Load two matrix from two file(A.csv, B.csv)
  - Compute multiplication of two matrix
  - Print the result matrix of this matrix multiplication

Two examples of 3x3 matrix



File: A.csv

```
3,3
48,69,8
24,79,24
32,59,10
```

File: B.csv

```
3,3
18,26,47
90,31,36
2,47,98
```