And You Thought There Couldn't be More C++

Outline

- Multi-File Programs
- makefiles

Multi-File Programs

- Advantages
  - If you write classes in separate files you can use those classes in multiple programs
    - Increases reusability
  - If you want to change something in a class, change it in one place and all programs using it reflect change
  - If multiple people are working on a project, makes sense to have separate files
    - Real world – many programmers will be working on one project

Multi-File Programs

- Class Libraries
  - We have included header files
    - These refer to files in the C++ library
  - Advantages
    - Reuse of code written by (and tested by) other people
  - You can write your own libraries if you like

Multi-File Programs

- Header files
  - Contain information about classes and functions
  - Usually .h extension – for “header”
  - Can then use the preprocessor directive #include to use them
    - e.g. #include <stdio.h>
    - stdio is a library of i/o files, stdio.h is the header file that describes all the available functions
  - Essentially describes the API to the functions and classes

Multi-File Programs

```
#include "Mathematics.h"
int Mathematics::add(int num1, int num2) {
        return Mathematics::result = num1 + num2;
}
int Mathematics::subtract(int num1, int num2) {
        return Mathematics::result = num1 - num2;
}
int Mathematics::multiply(int num1, int num2) {
        return Mathematics::result = num1 * num2;
}
int Mathematics::divide(int num1, int num2) {
        return Mathematics::result = num1 / num2;
}
```

Mathematics.cpp
source file

```
#ifndef MATHEMATICS_H
#define MATHEMATICS_H
#include <iostream>
class Mathematics
{
        int result;
        public:
        int add(int num1,int num2);
        int subtract(int num1,int num2);
        int multiply(int num1,int num2);
        int divide(int num1,int num2);
};
#endif
```

Mathematics.h
header file

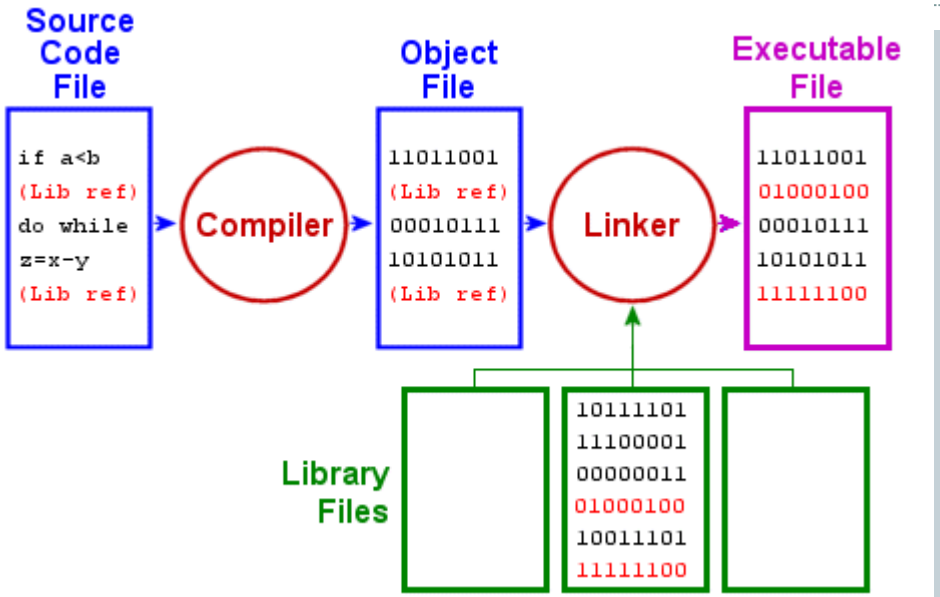
Multi-File Programs

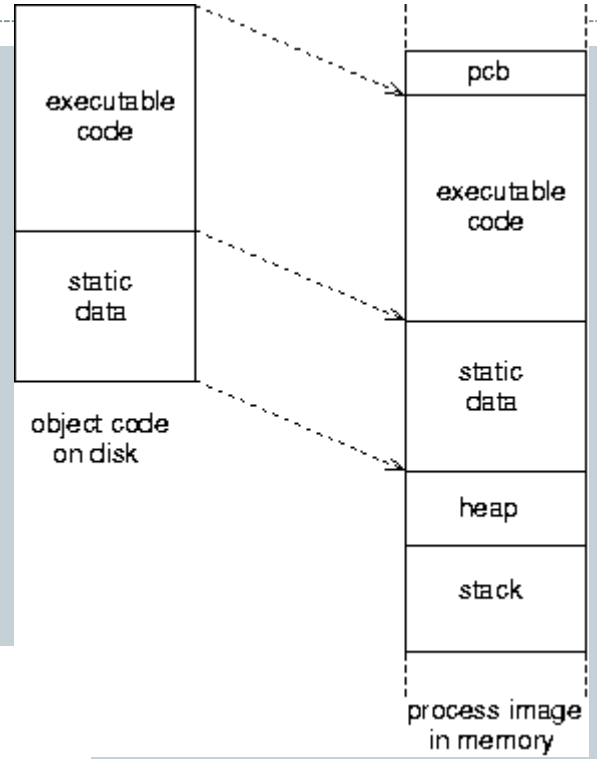
```
#include <iostream>
#include <string>
using namespace std;
#include "Mathematics.h"
int main() {
        int num1, num2, result;
        Mathematics maths;
        cout <<"Enter the first number:";
        cin>>num1;
        cout<<"Enter the 2nd number:";
        cin>>num2;
        result = maths.add(num1, num2);
        cout <<"\nThe result of adding two numbers is: "<<result<<endl;
        result = maths.subtract(num1, num2);
        cout <<"The result of subtracting two numbers is: "<<result<<endl;
        result = maths.multiply(num1, num2);
        cout <<"The result of multipltying two numbers is: "<<result<<endl;
        result = maths.divide(num1, num2);
        cout <<"The result of dividing two numbers is: "<<result<<endl;
}
```

Multi-File Programs

- Compiling
  - Let's say you have three .cpp source files, rectangle.cpp, ellipse.cpp, and main.cpp
  - One approach to compilation is:
    - g++ -c rectangle.cpp
    - g++ -c ellipse.cpp
    - g++ -c main.cpp
    - g++ -o myprogram rectangle.o ellipse.o main.o
  - You need not specify the header files because these will be #include(d) in the .cpp files

Compile vs. Link






```
#ifndef MATHEMATICS_H
#define MATHEMATICS_H
#include <iostream>
class Mathematics
{
        int result;
        public:
        int add(int num1,int num2);
        int subtract(int num1,int num2);
        int multiply(int num1,int num2);
        int divide(int num1,int num2);
};
#endif
```

Summary

- Multi-File Programs
- makefiles