

# MORE FUNCTIONS MAKEFILES

---

Problem Solving with Computers-I

**C++**

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

int main()
cout<<"Hola Facebook!";
return 0;
}
```



# Approximate Equality

What is the correct way to test for approximate equality?

(a)  $|x - y| < 0.001$

(b)  $|y - x| \leq 0.001$

(c)  $|x + y| < |x + y| + 0.001$

(d) A and B

(e) A, B, and C

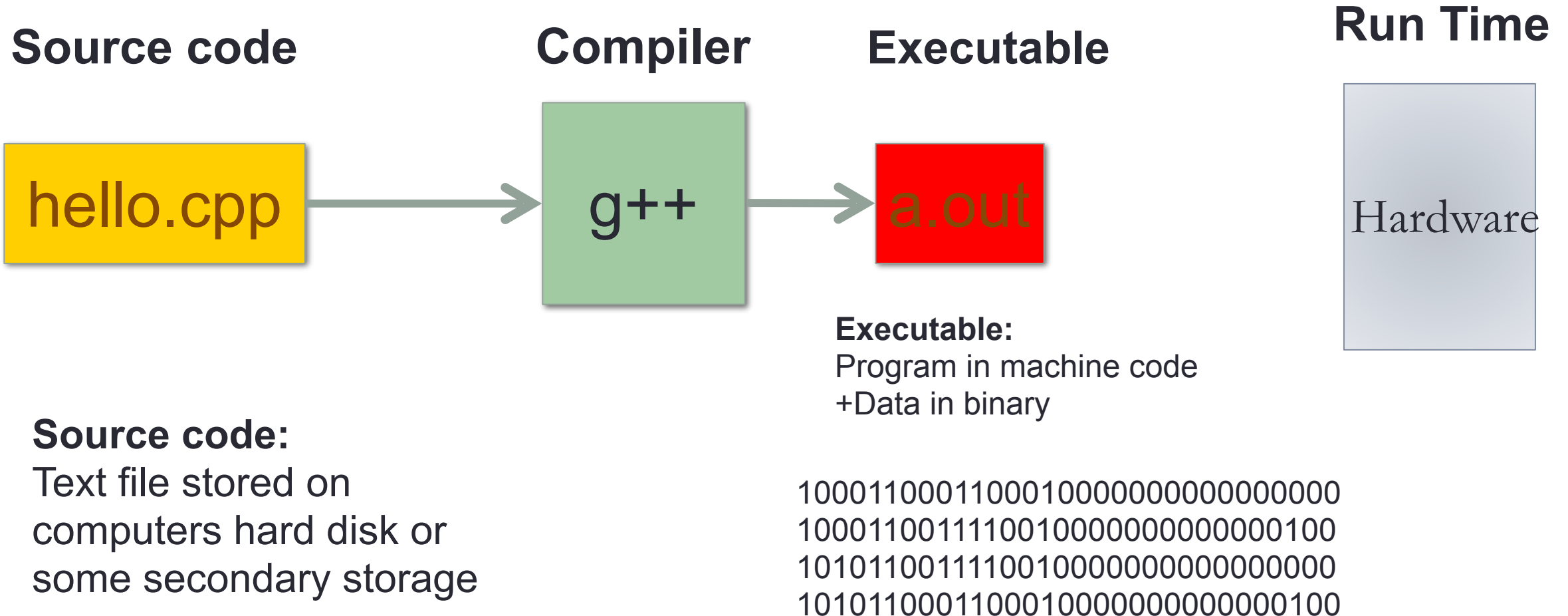
# Writing code that works

Write a function that RETURNS a string representing an isosceles triangle with a given width

```
s = drawTriangle(5);  
cout<<s;
```

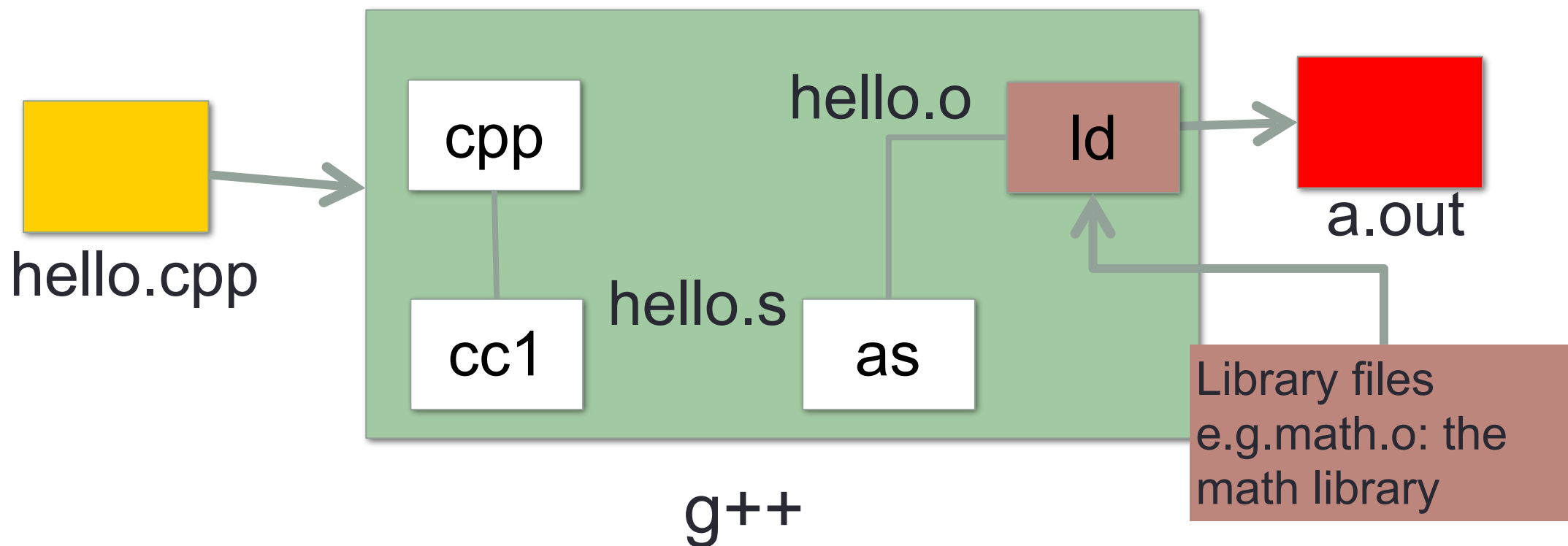
```
  *  
 ***  
*****
```

# The compilation process



# g++ is composed of a number of smaller programs

- Code written by others (libraries) can be included
- ld (linkage editor) merges one or more object files with the relevant libraries to produce a single executable



# Steps in gcc

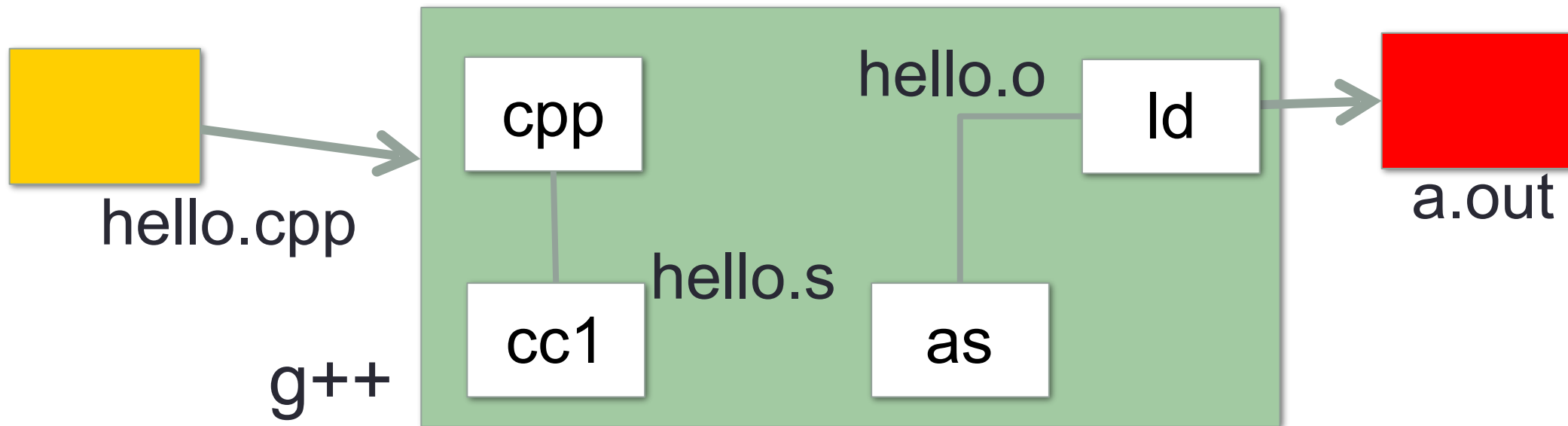
- Ask compiler to show temporary files:

```
$ g++ -S hello.cpp
```

```
$ g++ -c hello.o
```

```
$ g++ -o hello hello.cpp
```

```
$ g++ functions.o main.o -o myhello
```



# Make and makefiles

- The unix make program automates the compilation process as specified in a Makefile
- Specifies how the different pieces of a program in different files fit together to make a complete program
- In the makefile you provide a recipe for compilation
- When you run make it will use that recipe to compile the program

```
$ make
```

```
g++ testShapes.o shapes.o tdd.o -o testShapes
```

# Demo

- Basics of code compilation in C++ (review)
- Makefiles (used to automate compilation of medium to large projects) consisting of many files
- We will start by using a makefile to compile just a single program
- Extend to the case where your program is split between multiple files
- Understand what each of the following are and how they are used in program compilation
  - Header file (.h)
  - Source file (.cpp)
  - Object file (.o)
  - Executable
  - Makefile
  - Compile-time errors
  - Link-time errors



# Specifying a recipe in the makefile

- **Comments** start with a #
- **Definitions** typically are a variable in all caps followed by an equals sign and a string, such as:

```
CXX=g++  
CXXFLAGS=-Wall  
  
BINARIES=proj1
```

```
# testShapes is the target - it is what we want to produce  
# To produce the executable testShapes we need all the .o files  
# Everything to the right of ":" is a dependency for testShapes
```

```
testShapes: testShapes.o shapes.o tdd.o
```

```
    #The recipe for producing the target (testshapes) is below
```

```
g++ testShapes.o shapes.o tdd.o -o testShapes
```

# Print vs return