C++ Program Design
-- Supplement

Junjie Cao @ DLUT

Summer 2018

http://jjcao.github.io/cPlusPlus

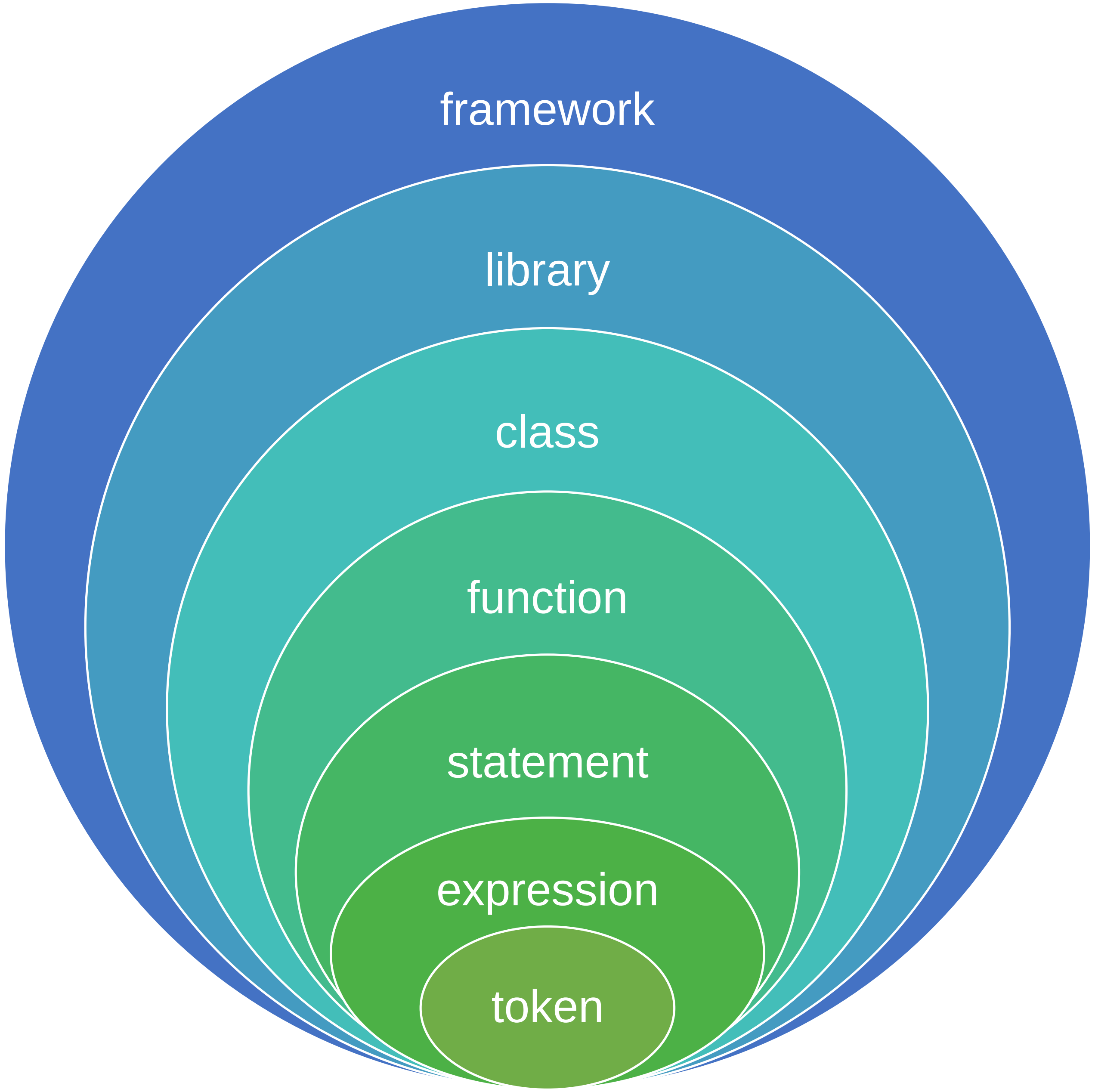
程序的结构

- Token标记, word,
- a = i+++j;

  a = i + (++j) ;

  a = (i++) + j ;
- Longest token possible, left-to-right

- Expression表达式, phrase, value
- Statement语句, sentence, ;
- Reuse begin with function & class;



Variables, Left & Right Value

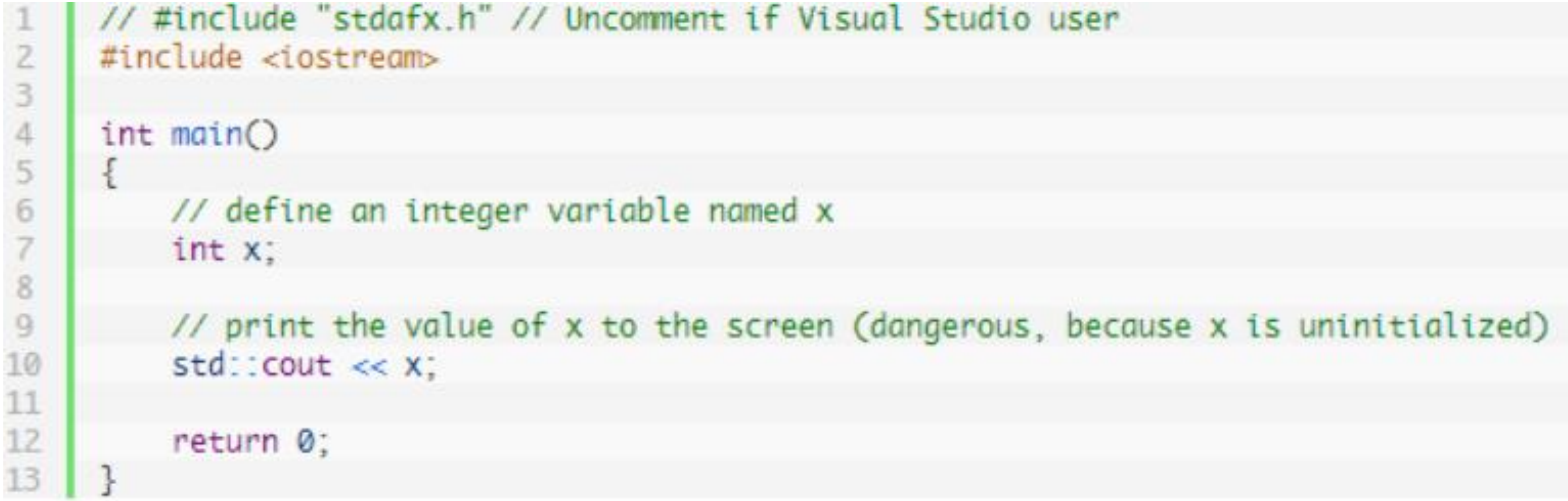
- A variable is a **named location in memory**
  - int x = 5;

- An **l-value** is a value that has an **address** (in **memory**).
- An **r-value** refers to any value that can be assigned to an l-value.
  - single numbers (such as 5, which evaluates to 5),
  - variables (such as x, which evaluates to whatever value was last assigned to it),
  - expressions (such as 2 + x, which evaluates to the value of x plus 2).

- Examples
  - while (x=1) y++;
  - while (x==1) y++;

Uninitialized variables未初始化的变量

- 可能导致不可预料的错误:

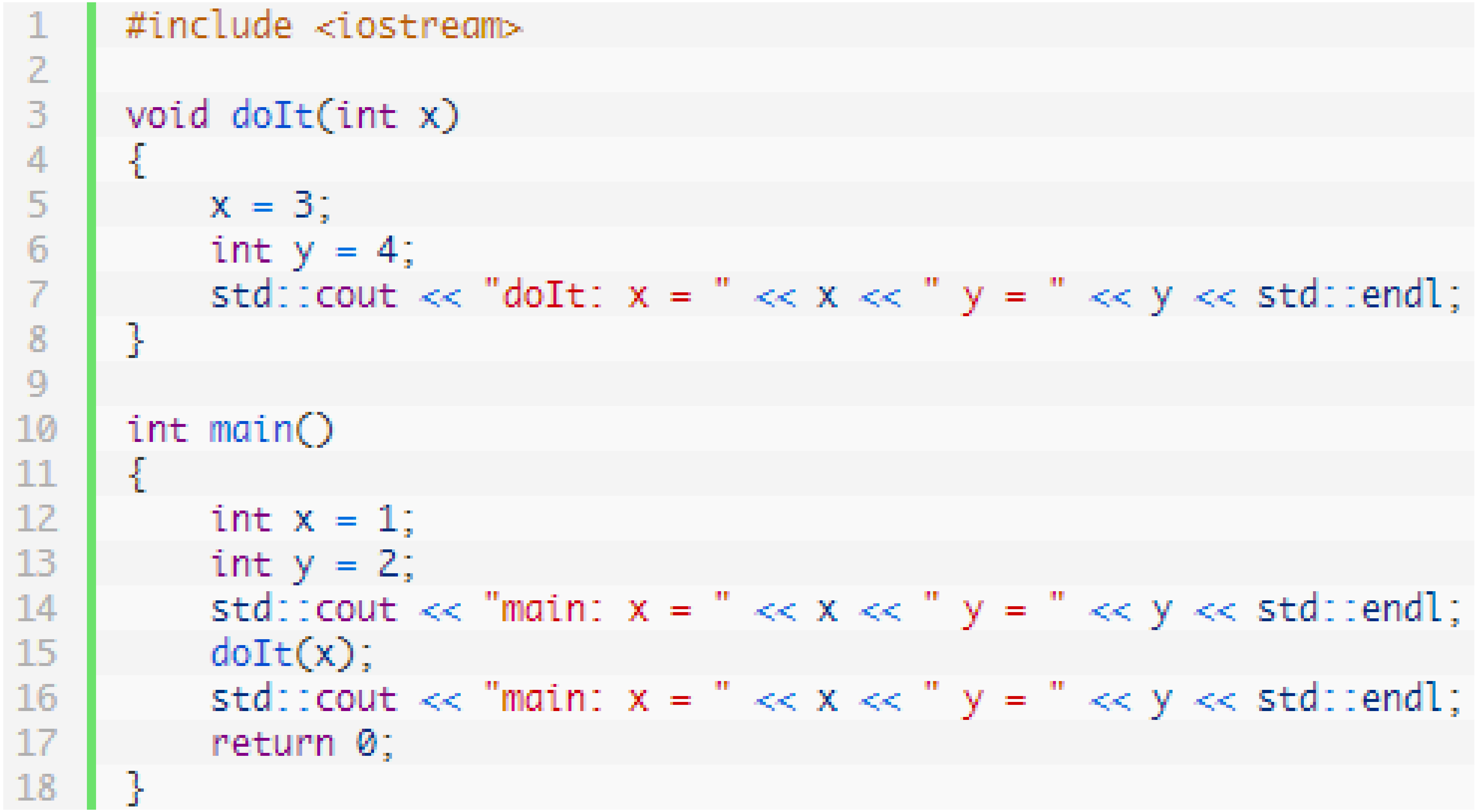


```
// #include "stdafx.h" // Uncomment if Visual Studio user
#include <iostream>

int main()
{
    // define an integer variable named x
    int x;

    // print the value of x to the screen (dangerous, because x is uninitialized)
    std::cout << x;

    return 0;
}
```

- X占用了一块无主地内存，不知道里面放着什么内容。
- 因此无预测打印的内容，每次运行这段程序的结果可能各不相同

局部范围防止名称冲突

下边程序会打印什么？

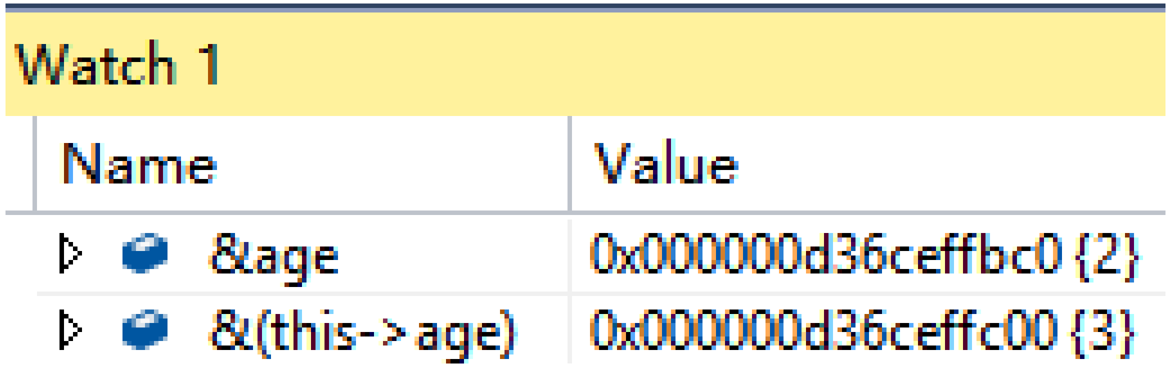


```cpp
#include <iostream>

void doIt(int x)
{
    x = 3;
    int y = 4;
    std::cout << "doIt: x = " << x << " y = " << y << std::endl;
}

int main()
{
    int x = 1;
    int y = 2;
    std::cout << "main: x = " << x << " y = " << y << std::endl;
    doIt(x);
    std::cout << "main: x = " << x << " y = " << y << std::endl;
    return 0;
}
```

名称冲突

```
class Student{
string id; int age;

Student(string id, int age):id(id),age(3){
cout << age << ‘,’;
age = age;
cout << age << ‘,’;
age = 3;
cout << age << ‘,’;
}
};
int age(2);
Student stu1("dd", age);
cout << age << endl;
```

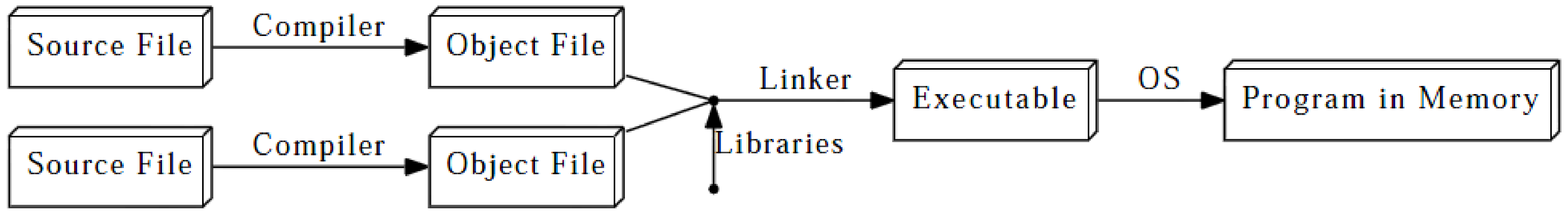
- 打印什么?
- 2, 2, 3, 2



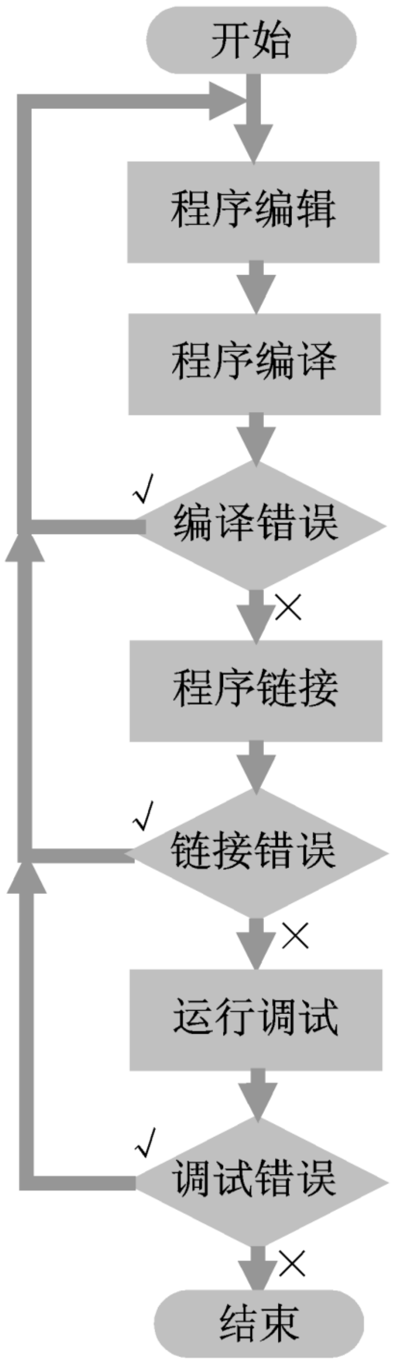
```
Watch 1
Name              Value
▷ &age            0x000000d36ceffbc0 {2}
▷ &(this->age)    0x000000d36ceffc00 {3}
```

&age 0x0000004a42aff8f4 {2}

The Building Process



compiler, link and run time errors!



Declarations, definitions声明和定义

- 以下代码会导致编译compile or 链接link or 编译和链接错误？

```cpp
#include <iostream>
int add(int x, int y);

int main()
{
    using namespace std;
    cout << "3 + 4 + 5 = " << add(3, 4, 5) << endl;
    return 0;
}

int add(int x, int y, int z)
{
    return x + y + z;
}
```

Quiz

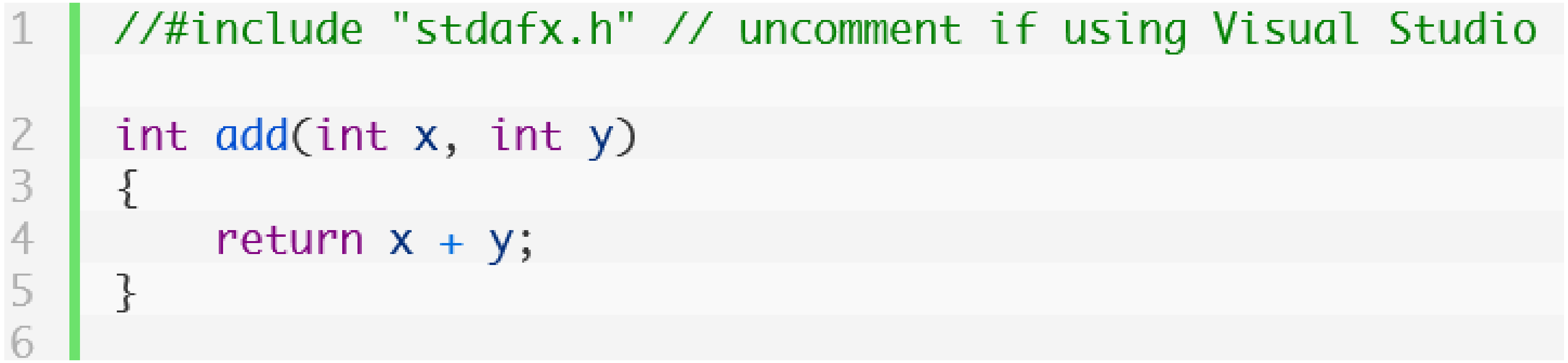
```cpp
#include <iostream>
int add(int x, int y);

int main()
{
    using namespace std;
    cout << "3 + 4 + 5 = " << add(3, 4) << endl;
    return 0;
}

int add(int x, int y, int z)
{
    return x + y + z;
```

```
Compiling...
add.cpp
Linking...
add.obj : error LNK2001: unresolved external symbol "int __cdecl add(int,int)" (?add@@YAHHH@Z)
add.exe : fatal error LNK1120: 1 unresolved externals
```

Programs with multiple files

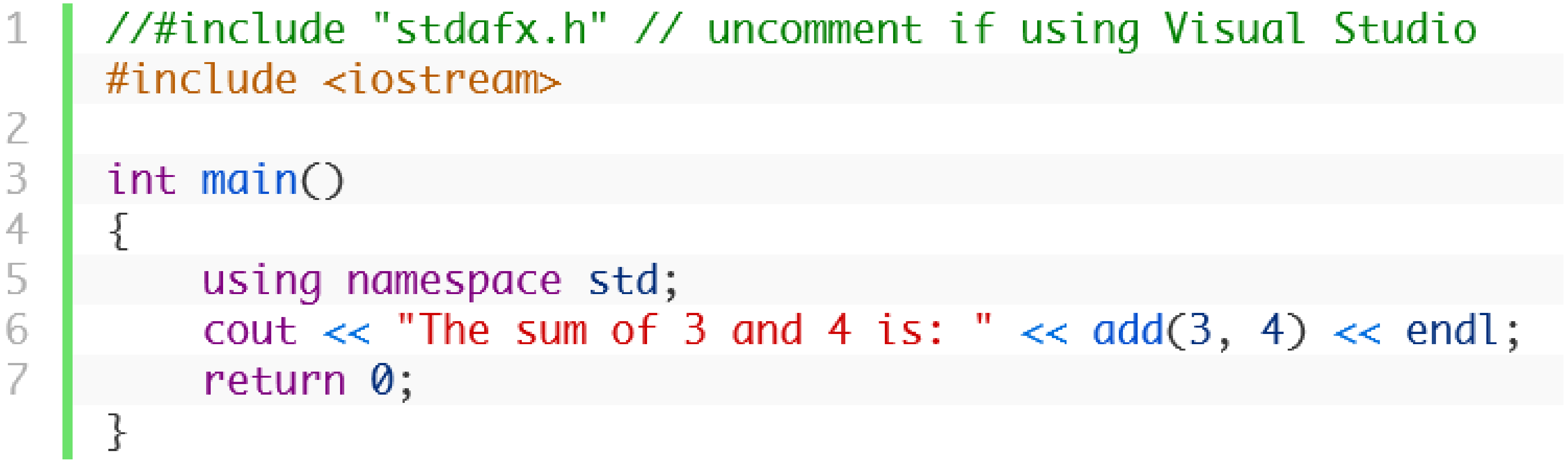
A multi-file example

- add.cpp:



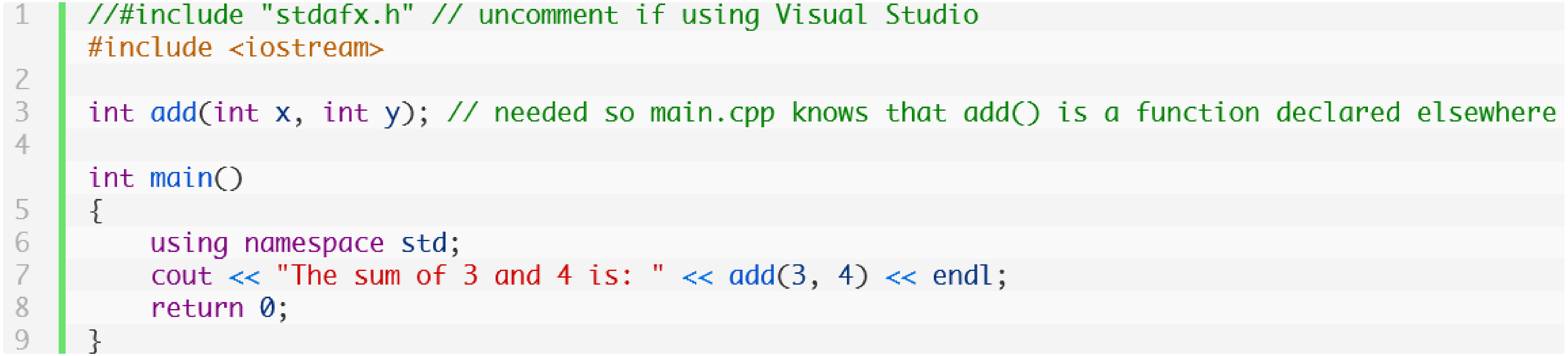
```cpp
//#include "stdafx.h" // uncomment if using Visual Studio

int add(int x, int y)
{
    return x + y;
}

```

- main.cpp:



```cpp
//#include "stdafx.h" // uncomment if using Visual Studio
#include <iostream>

int main()
{
    using namespace std;
    cout << "The sum of 3 and 4 is: " << add(3, 4) << endl;
    return 0;
}
```

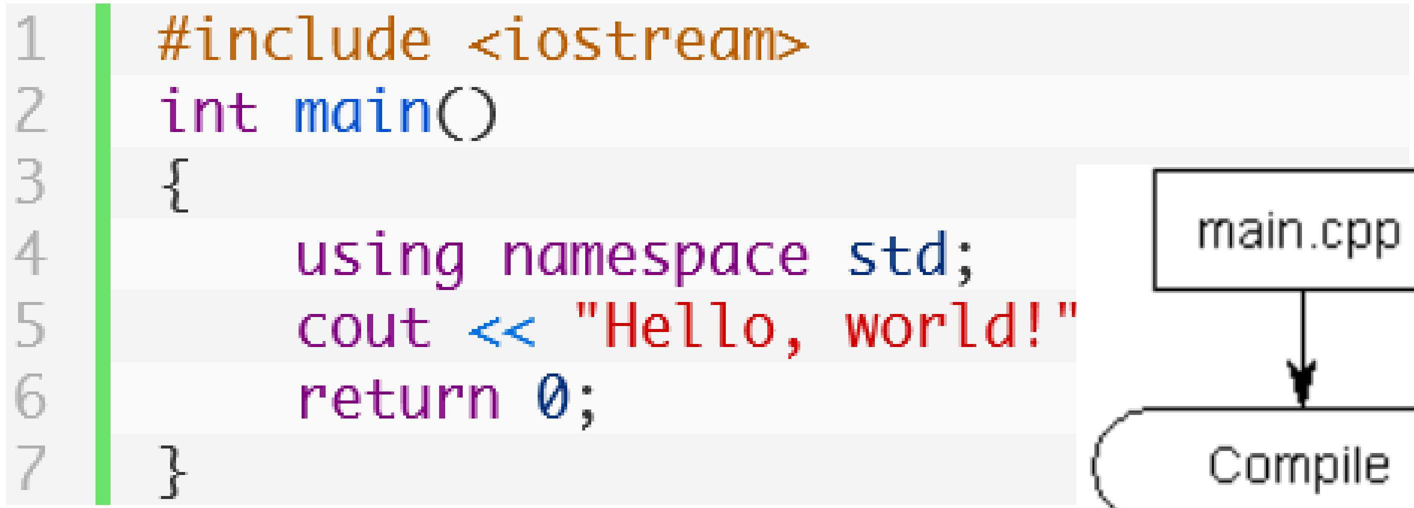
main.cpp:



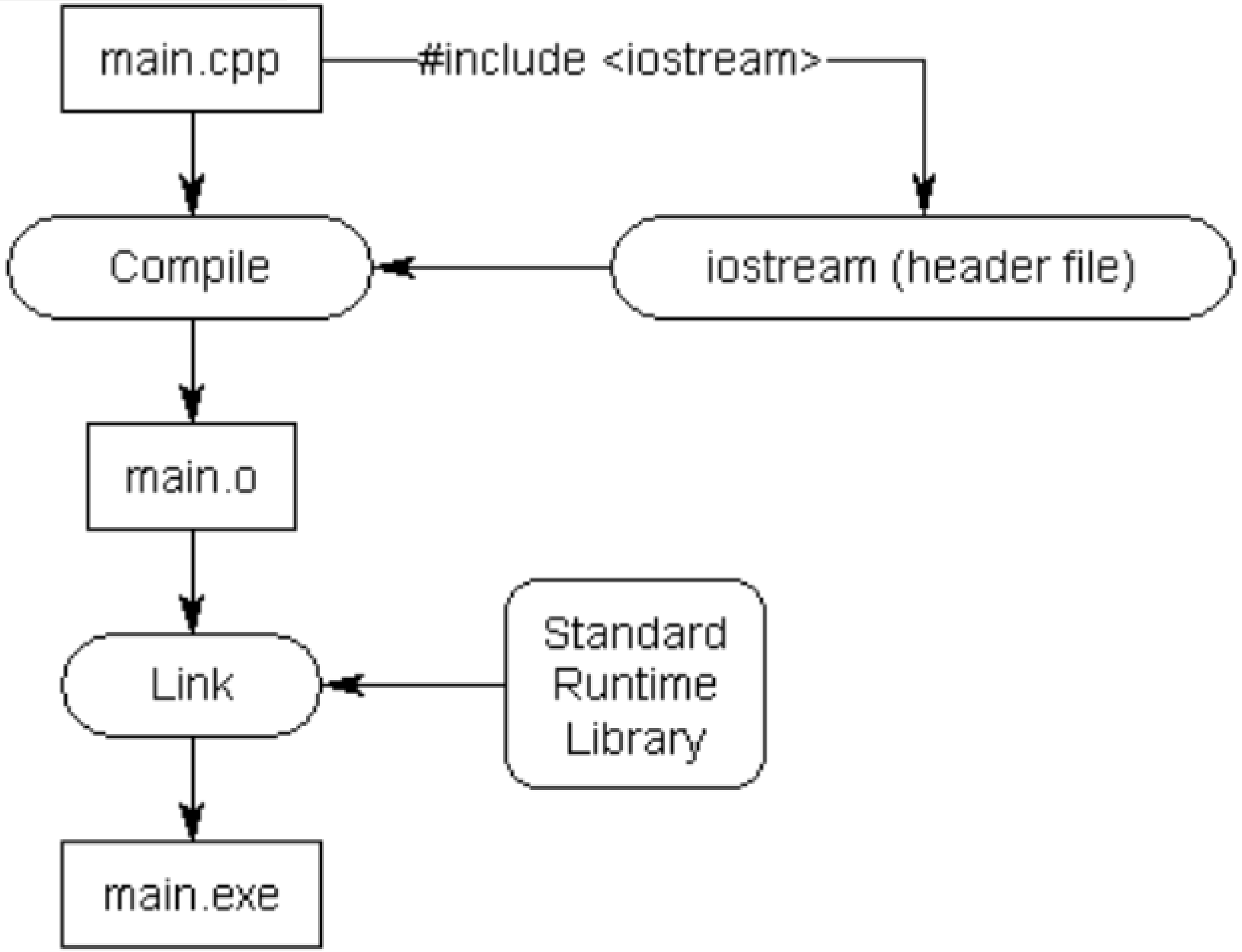
```cpp
//#include "stdafx.h" // uncomment if using Visual Studio
#include <iostream>

int add(int x, int y); // needed so main.cpp knows that add() is a function declared elsewhere

int main()
{
    using namespace std;
    cout << "The sum of 3 and 4 is: " << add(3, 4) << endl;
    return 0;
}
```

- Does forward declaration前置声明 work?
- Yes

Using standard library header files

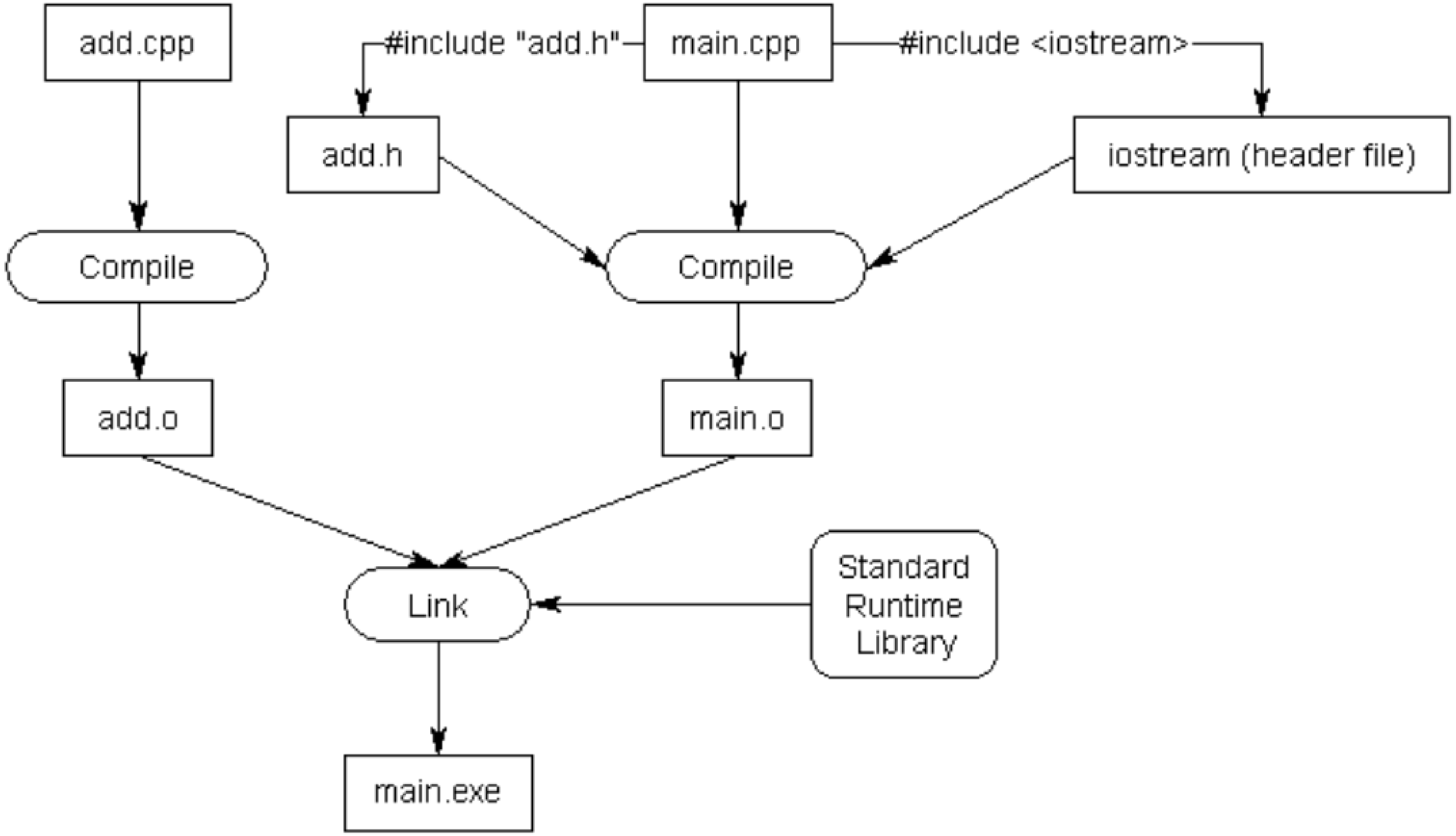


```
#include <iostream>
int main()
{
    using namespace std;
    cout << "Hello, world!"
    return 0;
}
```



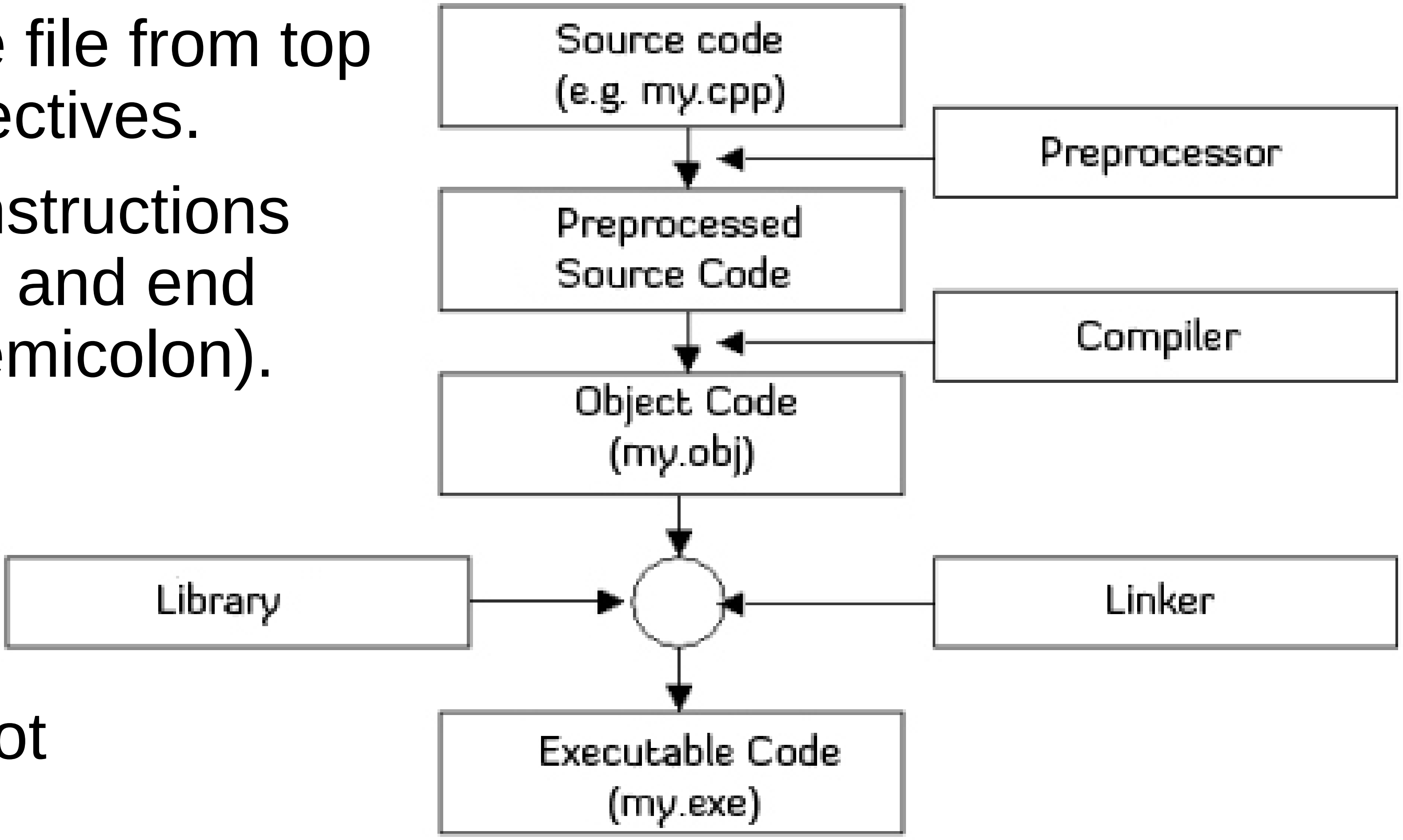
Use your own header files

- *When you #include a file, the entire content of the included file is inserted at the point of inclusion.*



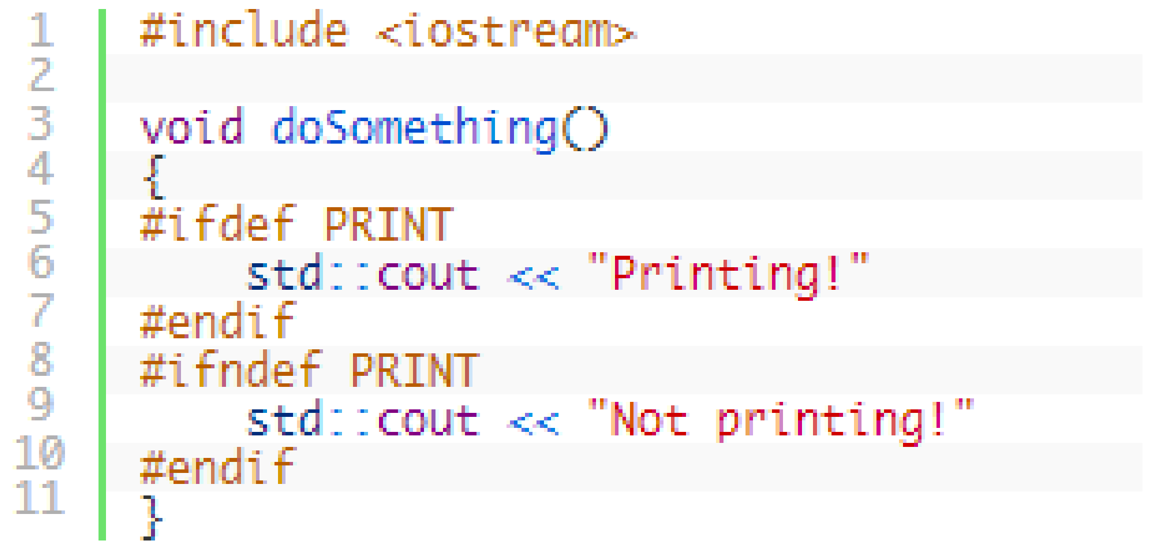
Preprocessor预处理器

- scans through each code file from top to bottom, looking for directives.
- **Directives** are specific instructions that start with a # symbol and end with a newline (NOT a semicolon).
- it is not smart -- it does not understand C++ syntax;
- simply manipulates text



Conditional compilation

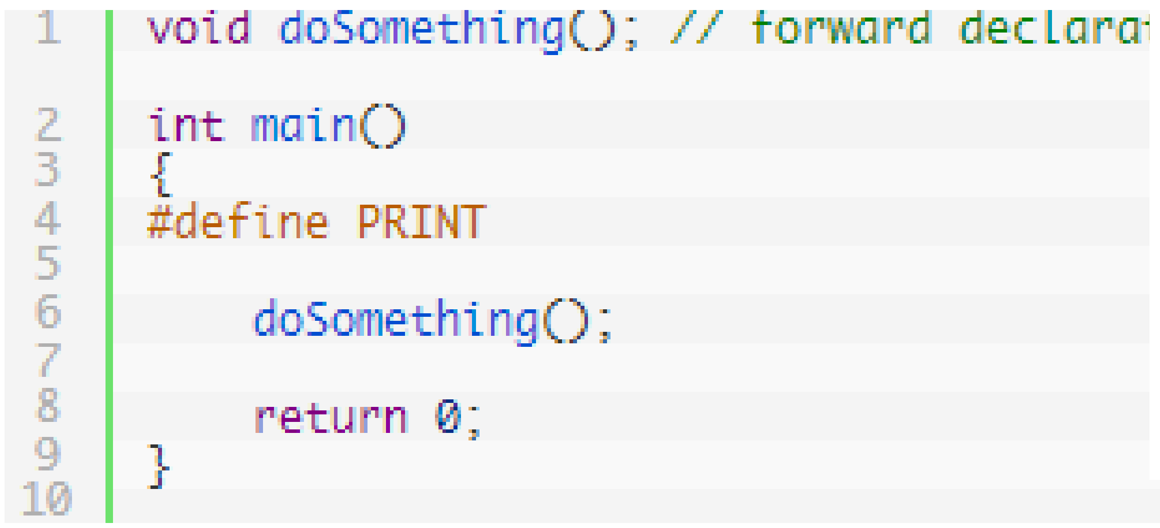
function.cpp:



```
#include <iostream>

void doSomething()
{
#ifdef PRINT
    std::cout << "Printing!"
#endif
#ifndef PRINT
    std::cout << "Not printing!"
#endif
}
```

Not printing!

main.cpp:



```
void doSomething(); // forward declarat

int main()
{
#define PRINT

    doSomething();

    return 0;
}
```

- Function.cpp => function.obj

  // sth from iostream

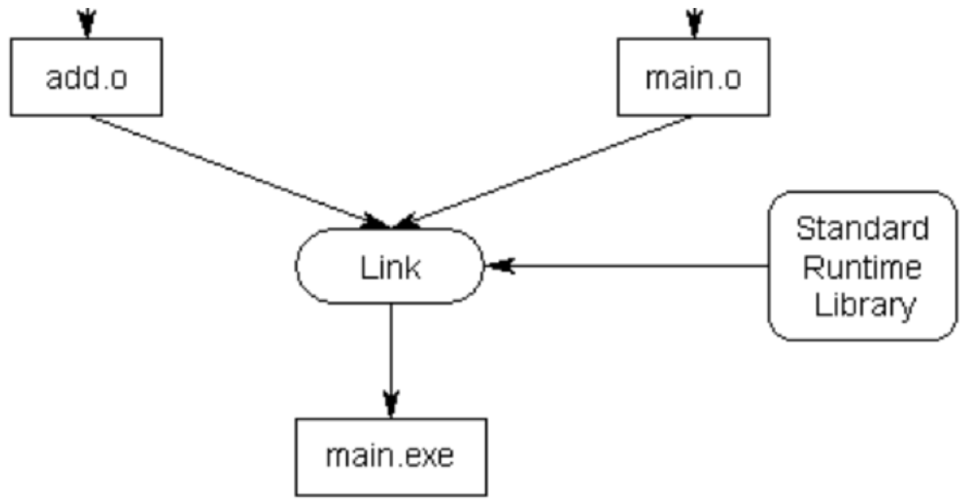
  void doSomething(){

  std::cout << “Not printing!”;}
- Main.cpp => main.obj

  void doSomething();

  int main(){ doSomething();

  Return 0;}

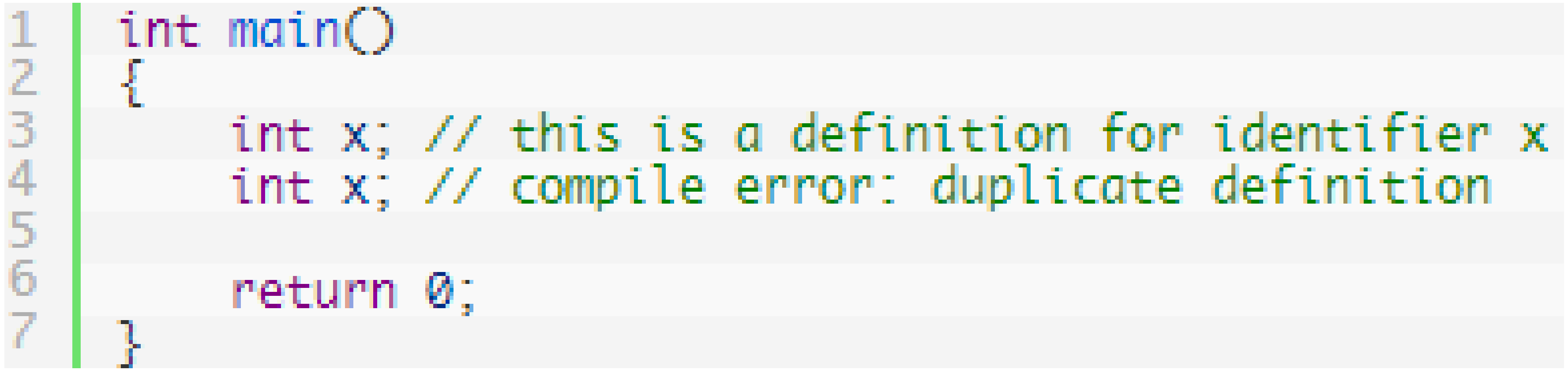


Header guards

a kind of conditional compilation

The duplicate definition problem

- an identifier can only have one definition



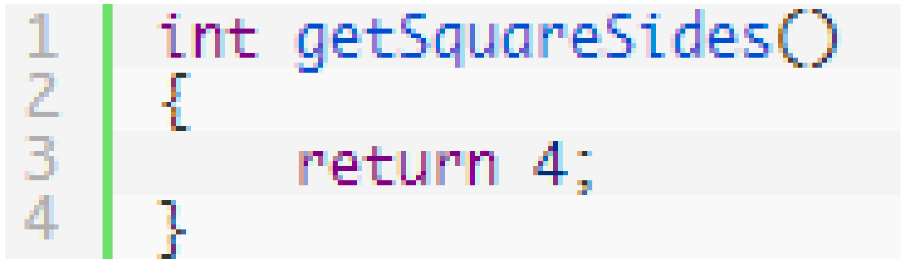
```cpp
int main()
{
    int x; // this is a definition for identifier x
    int x; // compile error: duplicate definition

    return 0;
}
```

The duplicate definition problem

- When a header file #includes another header file (which is common).
- How to resolve this issue?

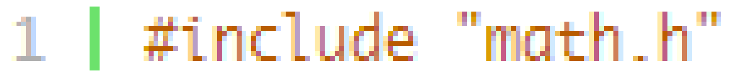
```
int getSquareSides(){// from math.h
    return 4;
}

int getSquareSides(){// from geometry.h
    return 4;
}

int main(){
    return 0;
}
```

math.h:



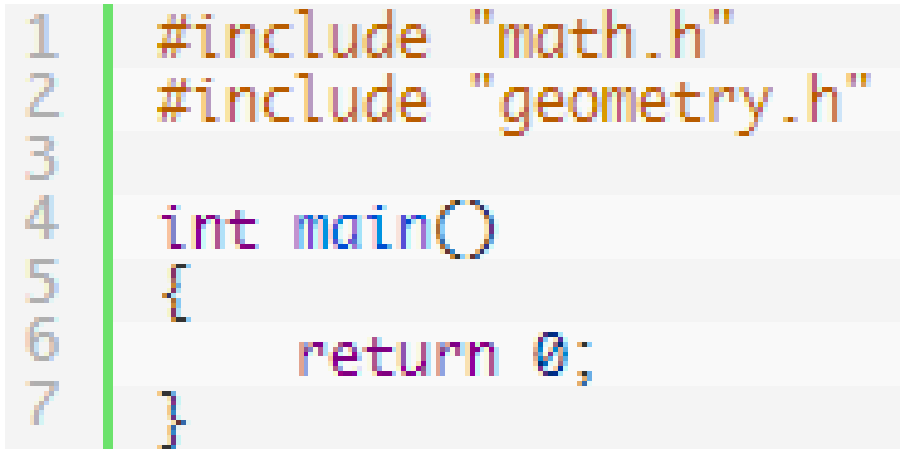
```
int getSquareSides()
{
    return 4;
}
```

geometry.h:



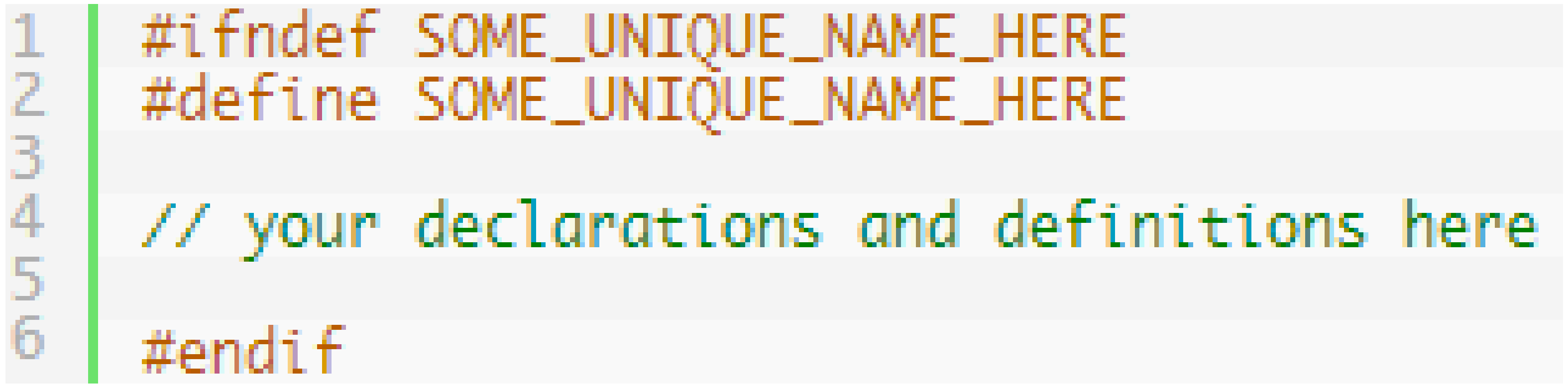
```
#include "math.h"
```

main.cpp:



```
#include "math.h"
#include "geometry.h"

int main()
{
    return 0;
}
```

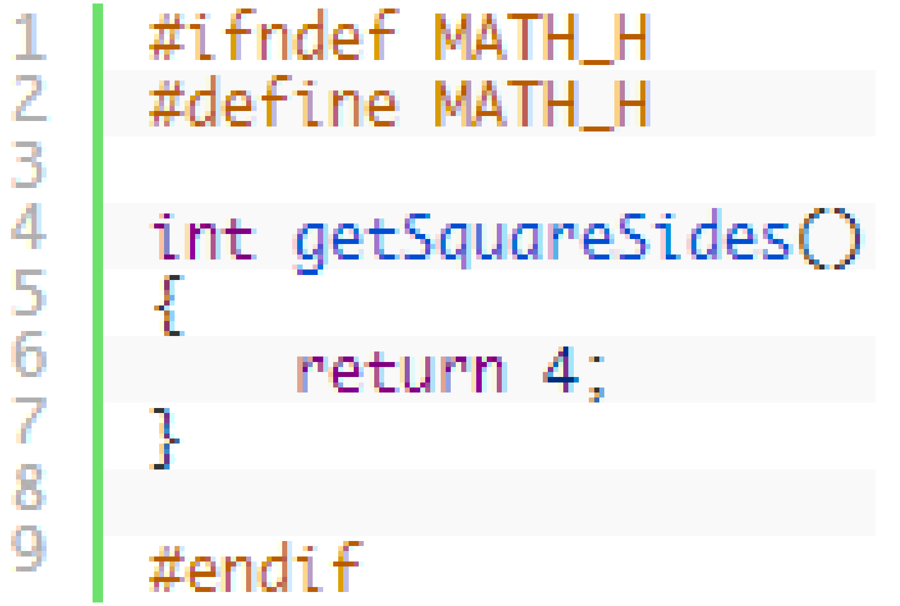
Header guards



```
#ifndef SOME_UNIQUE_NAME_HERE
#define SOME_UNIQUE_NAME_HERE

// your declarations and definitions here

#endif
```

- All header files should have header guards
- SOME_UNIQUE_NAME_HERE: typically the name of the header file with a _H appended to it

math.h:

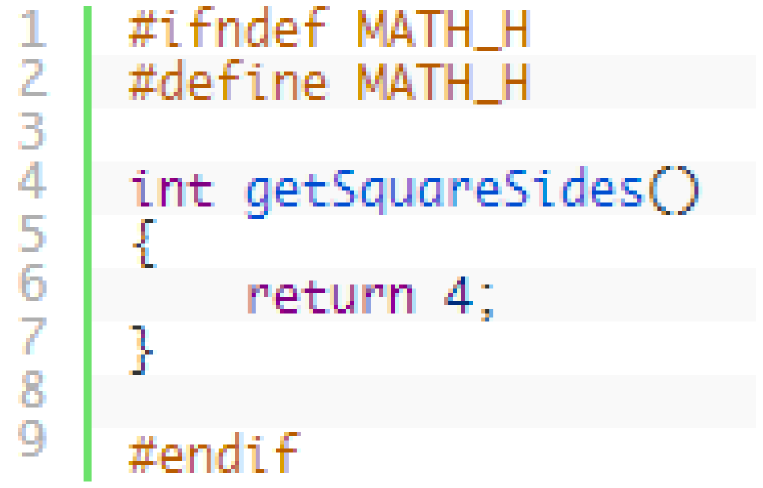


```
#ifndef MATH_H
#define MATH_H

int getSquareSides()
{
    return 4;
}

#endif
```

Updating our previous example with header guards

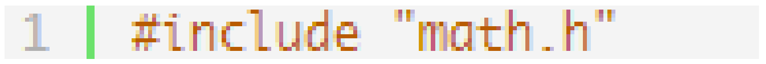
```
int getSquareSides(){// from math.h
    return 4;
}

// nothing from geometry.h

int main(){
    return 0;
}
```

math.h



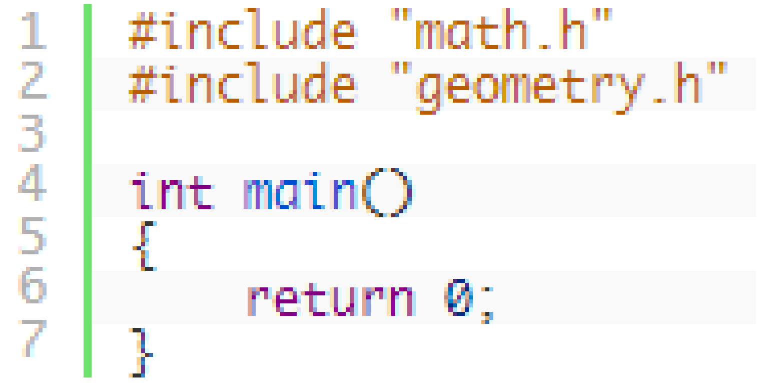
```
#ifndef MATH_H
#define MATH_H

int getSquareSides()
{
    return 4;
}

#endif
```

geometry.h:



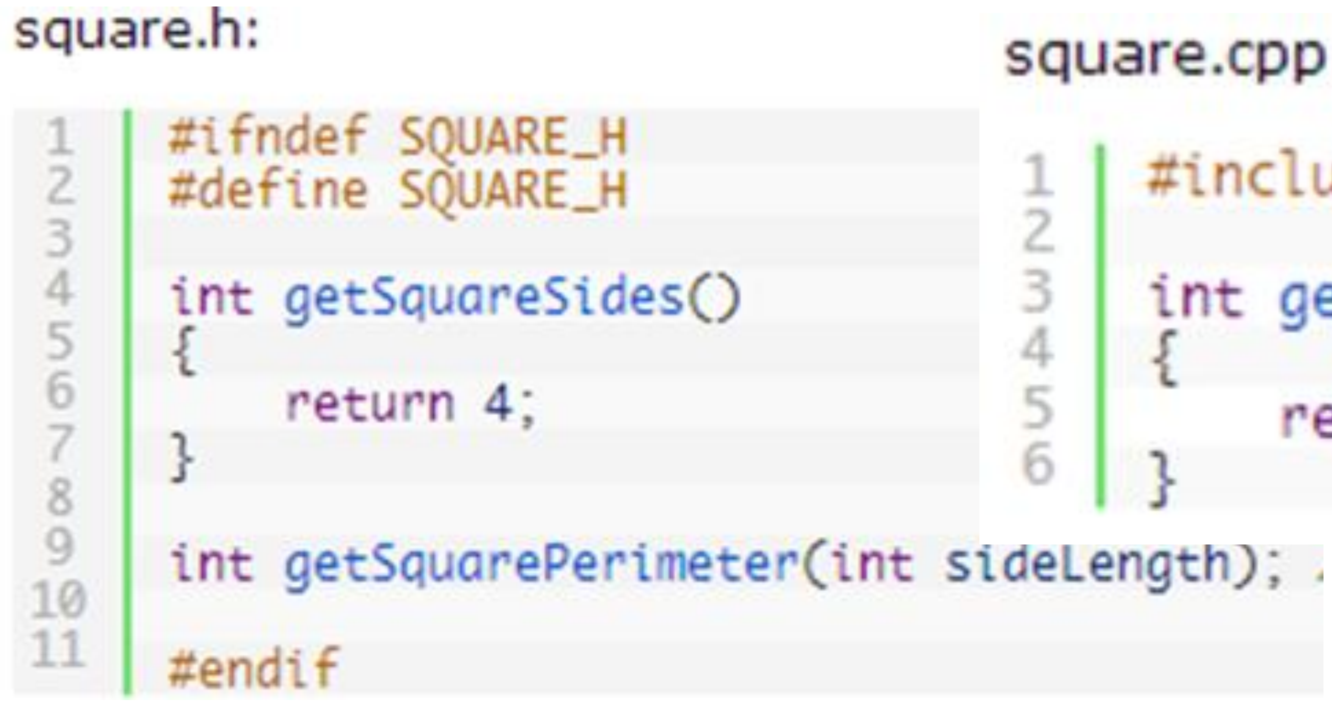
```
#include "math.h"
```

main.cpp:



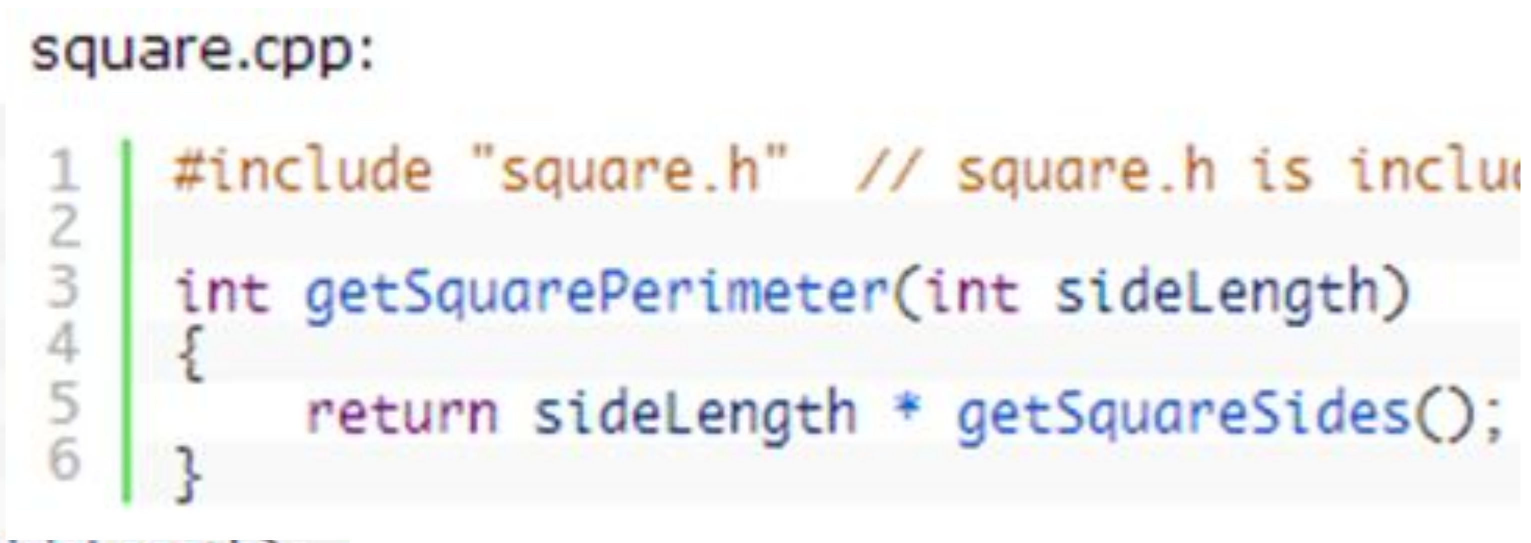
```
#include "math.h"
#include "geometry.h"

int main()
{
    return 0;
}
```

Header guards do not prevent a header from being included once into different code files



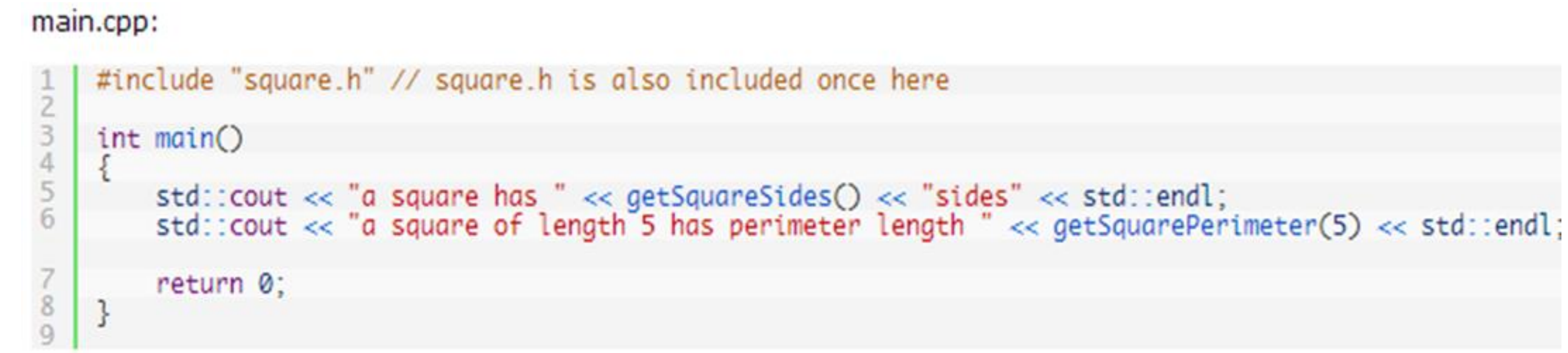
square.h:

```cpp
#ifndef SQUARE_H
#define SQUARE_H

int getSquareSides()
{
    return 4;
}

int getSquarePerimeter(int sideLength); /

#endif
```



square.cpp:

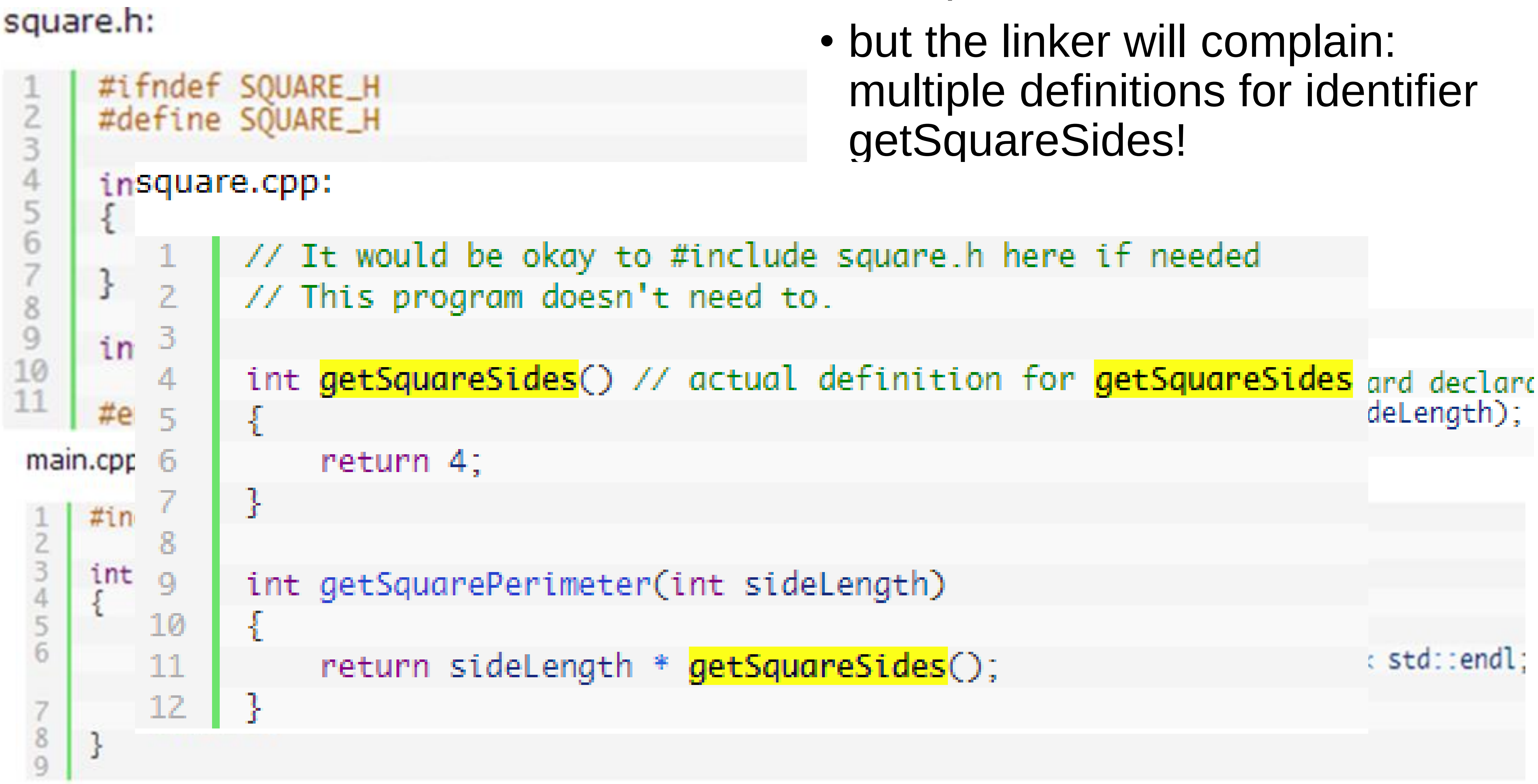
```cpp
#include "square.h"  // square.h is inclu

int getSquarePerimeter(int sideLength)
{
    return sideLength * getSquareSides();
}
```



main.cpp:

```cpp
#include "square.h" // square.h is also included once here

int main()
{
    std::cout << "a square has " << getSquareSides() << "sides" << std::endl;
    std::cout << "a square of length 5 has perimeter length " << getSquarePerimeter(5) << std::endl;

    return 0;
}
```

- Compile!
- but the linker will complain: multiple definitions for identifier getSquareSides!



square.h:

```
#ifndef SQUARE_H
#define SQUARE_H

in
{

}

in

#e
```

```
ard declar
deLength);
```

main.cpp

```
#in

int
{

                                    : std::endl;

}
```

square.cpp:

```
// It would be okay to #include square.h here if needed
// This program doesn't need to.

int getSquareSides() // actual definition for getSquareSides
{
    return 4;
}

int getSquarePerimeter(int sideLength)
{
    return sideLength * getSquareSides();
}
```

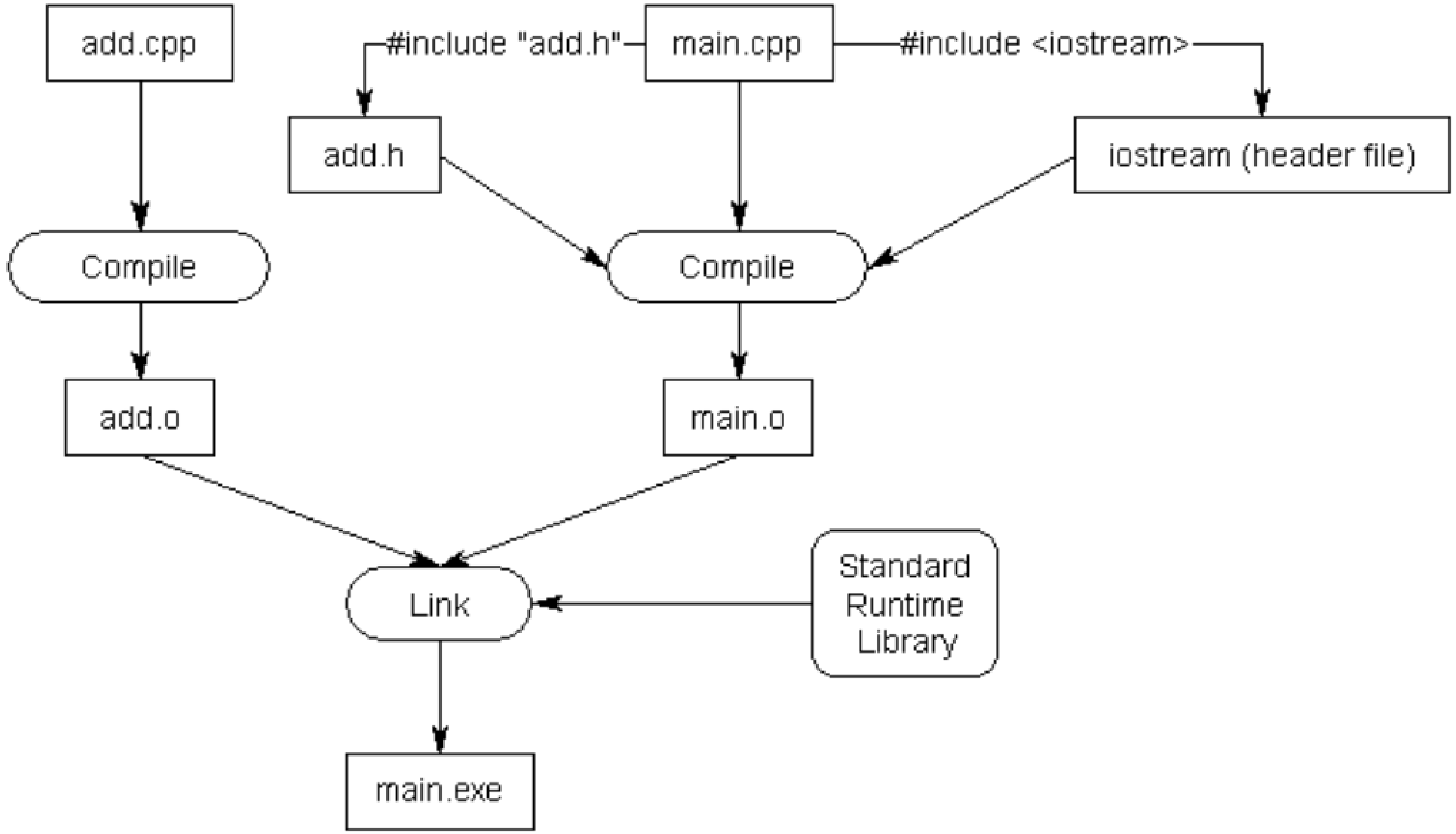
Header file best practices

1. Always include header guards.

2. Do not define variables in header files unless they are constants. Header files should generally only be used for declarations.

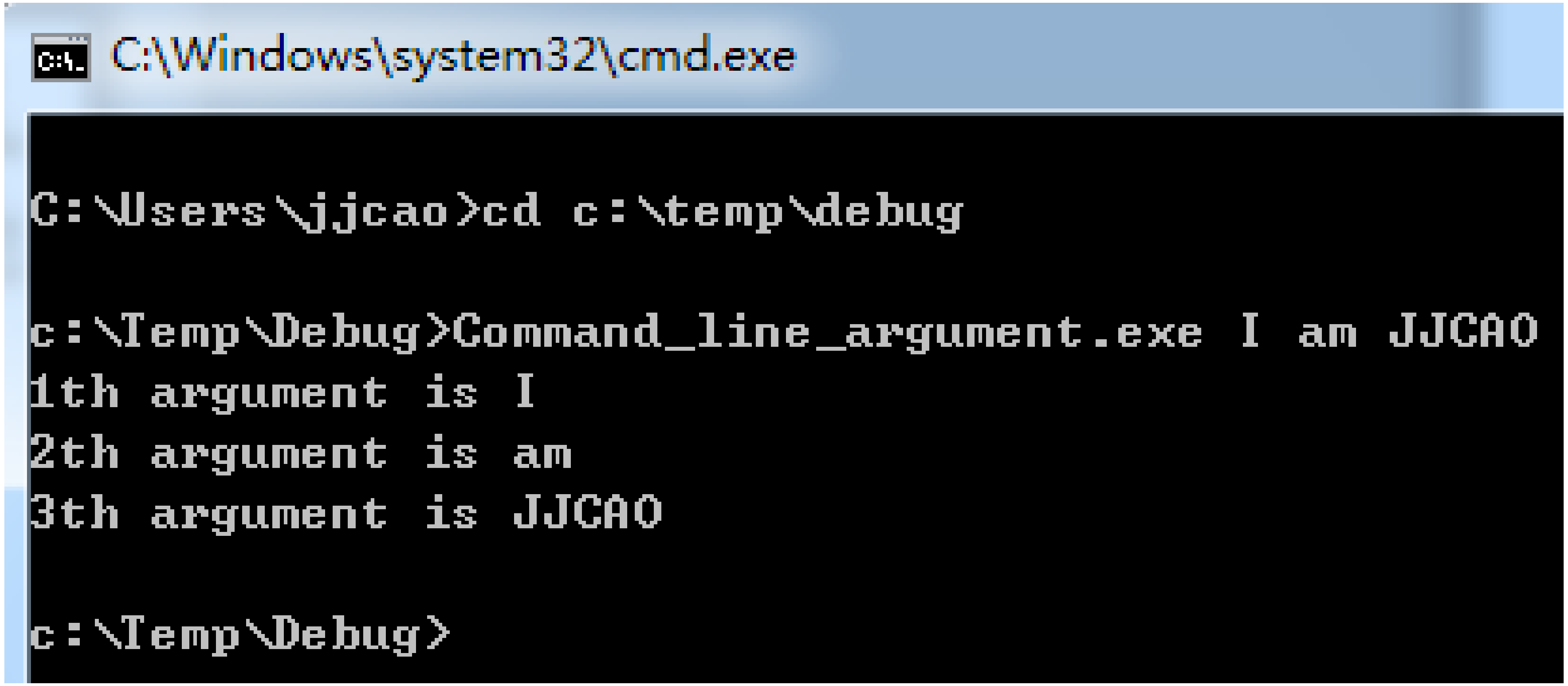
3. Do not define functions in header files.

反复再重复这个问题


add.cpp
#include "add.h"
main.cpp
#include <iostream>
add.h
iostream (header file)
Compile
Compile
add.o
main.o
Link
Standard Runtime Library
main.exe

Win32 Console Application

- CMD: command shell
- Command line arguments



```
C:\Users\jjcao>cd c:\temp\debug

c:\Temp\Debug>Command_line_argument.exe I am JJCAO
1th argument is I
2th argument is am
3th argument is JJCAO

c:\Temp\Debug>
```

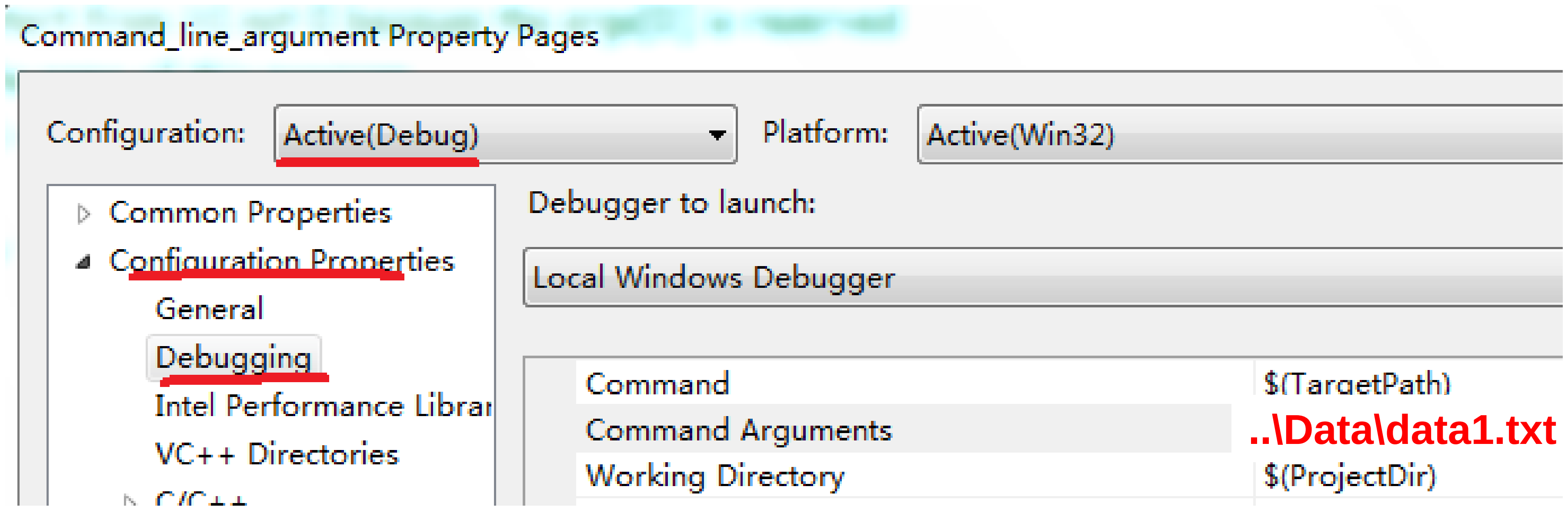
Files organization

- Solution1
  - Project1
    - Debug
      - p1.exe
    - Main.cpp; add.cpp, add.h;
    - Project1.dsp
  - Project2
    - Main.cpp
    - …
  - Debug
  - solution1.sln

- Solution1(续左)
  - Include
    - Lib1
      - Lib1.h
      - …
    - Lib2
  - Data
    - Data1.txt
    - …

How to read data1.txt using p1.exe?

- In command shell
  - c:/solution1/project1/debug> p1 ..\..\Data\data1.txt
- In Visual Studio



Command_line_argument Property Pages

Configuration: Active(Debug) Platform: Active(Win32)

Debugger to launch: Local Windows Debugger

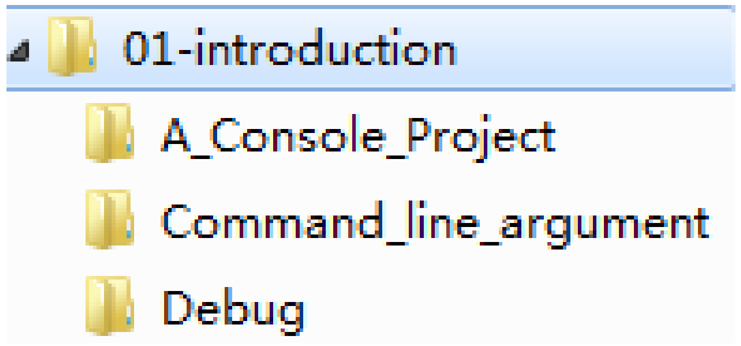
Command	$(TargetPath)
Command Arguments	..\Data\data1.txt
Working Directory	$(ProjectDir)

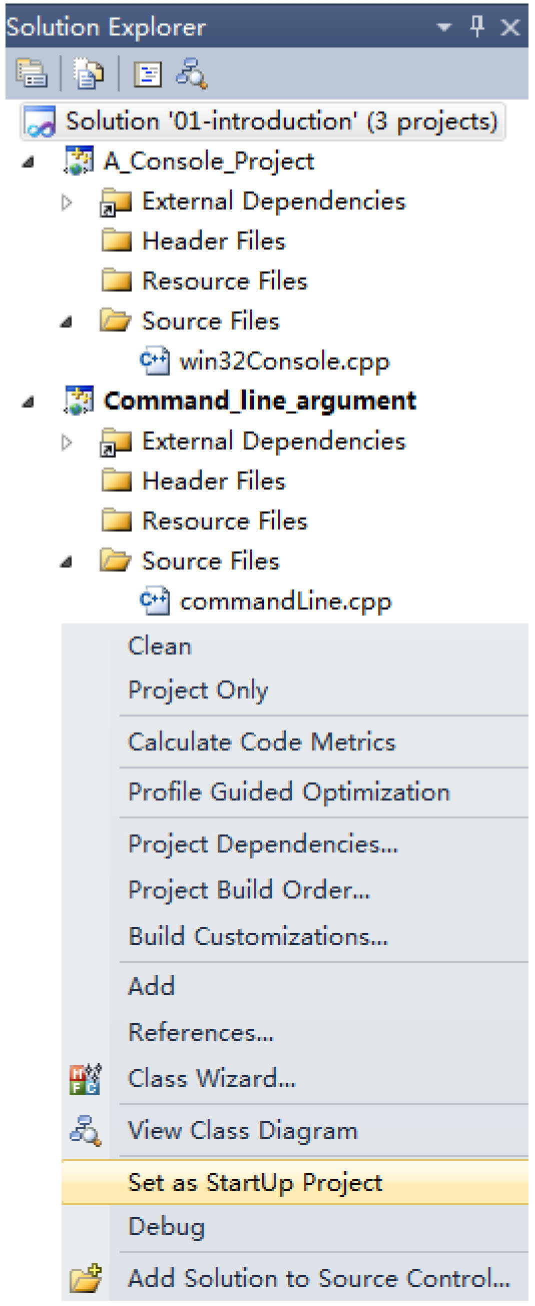
Solution1
  Project1
    Debug
      p1.exe
    Project1.dsp
  Data
    Data1.txt
    …

- **Current directory for executing your program**

Default current directory of VC

- **Solution**: 01-Introduction
  - Project: A_Console_Project
  - Project: Command_line_argument





- **Current Project**: Command_line_argument
- The **current directory** of the current project
  - The dir where the Command_line_argument.vcxproj is
  - Where is win32Console.cpp?
    ../ A_Console_Project/

How to include lib1.h in main.cpp?

- Current directory containing our source code first
  - **different** with current directory for executing your program

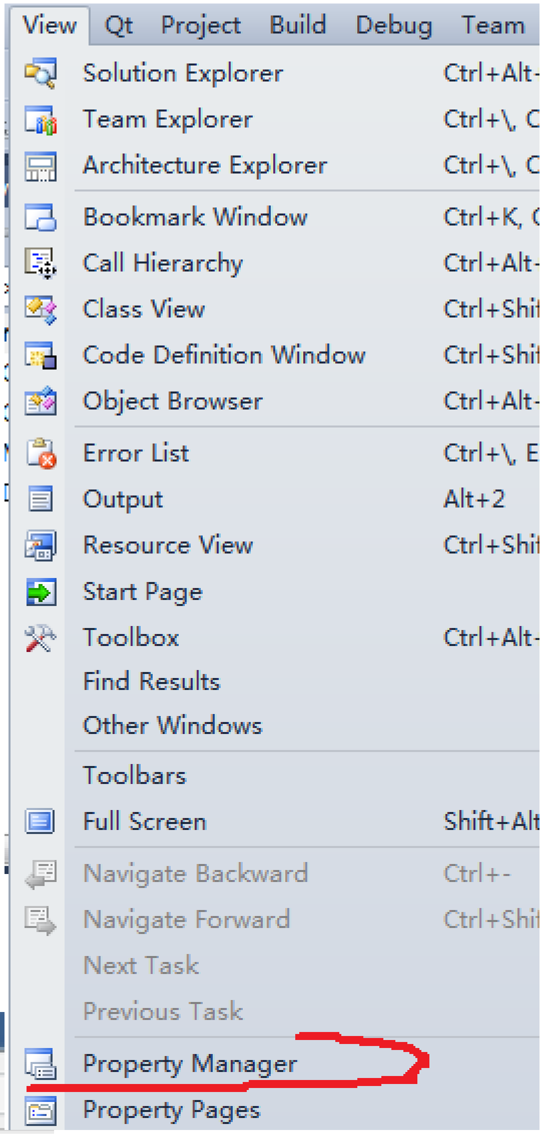
- #include "..\include\lib1.h"

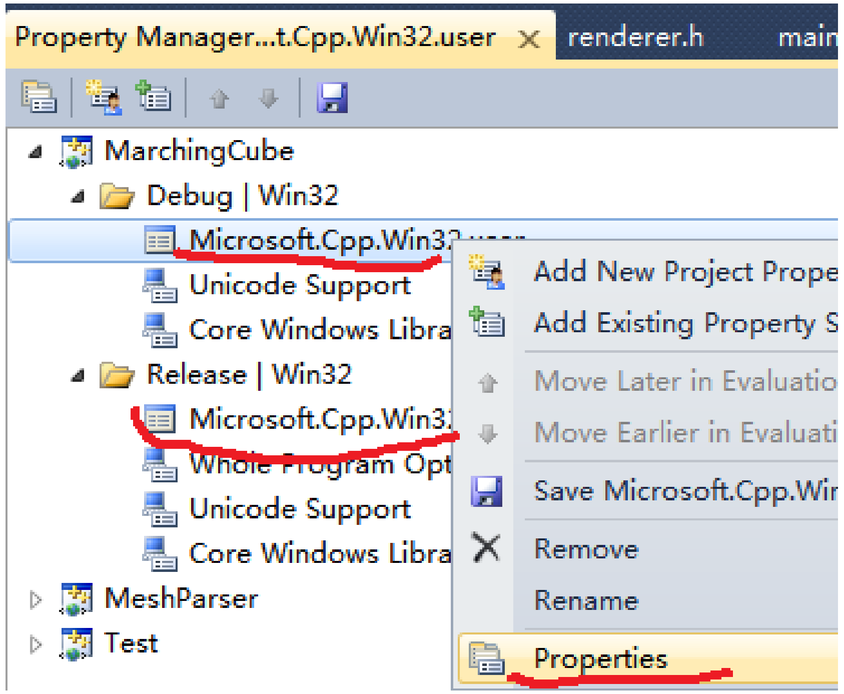
Solution1

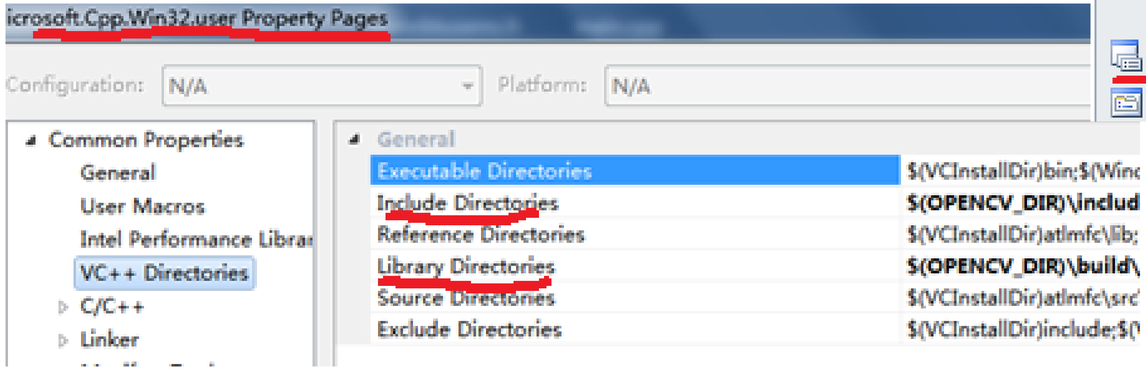
- Project1
  - Main.cpp;
- Include
  - Lib1
    - Lib1.h
    - …
  - Lib2
- Data
  - Data1.txt
  - …

Set include & lib path independent with solutions

Set it in Property Manager (You have to open a project first.) If you set it in the Context Menu of a solution or project, it will be dependent on specified projections.






icrosoft.Cpp.Win32.user Property Pages

Configuration: N/A Platform: N/A

General

Executable Directories	$(VCInstallDir)bin;$(Winc
Include Directories	$(OPENCV_DIR)\includ
Reference Directories	$(VCInstallDir)atlmfc\lib;
Library Directories	$(OPENCV_DIR)\build\
Source Directories	$(VCInstallDir)atlmfc\src
Exclude Directories	$(VCInstallDir)include;$(

Basic formatting

- Name
- Length
- Whitespace & alignment

Debugging your code

Six Stages of Debugging

1. That can't happen.
2. That doesn't happen on my machine.
3. That shouldn't happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?