03 Developing Programs

1 Compile a Program

We write C++ source code: **my_project.cpp**

Then we have **two ways** to compile it:

(1) g++ -o my_project my_project.cpp

(2) g++ -c my_project.cpp // To generate object code my_project.o

    g++ -o my_project my_project.o

Note: Object code (*.o) is not equivalent to executable code. Object code is a portion of machine code that hasn't yet been linked into a complete program.

2 Multiple Source Files

Two types of files:

(1) header files (.h) : class definitions and function declarations

```
//add.h
#ifndef ADD_H
#define ADD_H
int add(int a, int b);
#endif
```

(2) C++ source files (.cpp) : member functions of classes and function definitions

```
//add.cpp
int add(int a, int b)
{
    return a+b;
}
```

add.h and add.cpp complete a function add( )

If we want to use this function add( ) in another file (run_add.cpp), we should put #include "add.h":

```
//run_add.cpp
#include "add.h"
int main()
{
    add(2,3);
    return 0;
}
```

3 Header Guard

```
//add.h
#ifndef ADD_H // test whether ADD_H has not been defined before
#define ADD_H
int add(int a, int b);
#endif
```

Notes: If ADD_H has not been defined before, #ifndef succeeds and all lines up to #endif are processed. Otherwise, #ifndef fails and all lines between #ifndef and #endif are ignored.

What will happen for the following two header files, with/without header guard in add.h?

my_project1.h

```
#include "add.h"
...
```

my_project2.h

```
#include "add.h"
#include "my_project1.h"
```

4 Makefile

all: run_add

run_add: run_add.o add.o

  g++ -o run_add run_add.o add.o

run_add.o: run_add.cpp

  g++ -c run_add.cpp

add.o: add.cpp

  g++ -c add.cpp

clean:

  rm-f run_add *.o

general format:

Target: Dependency

 Command

Notes: -f: force to remove, even the objects are not there to avoid error

Type "make" on command-line for the first target ("all" in this case)

Type "make"for a specific

Example (windows):

simple_adder.h

```
#ifndef __SIMPLE_ADDER_H__
#define __SIMPLE_ADDER_H__
int add(int a, int b);
#endif
```

simple_adder.cpp

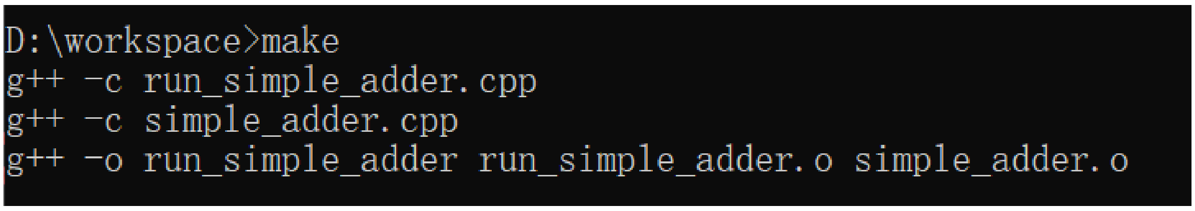
```
#include <iostream>
#include <string>
#include <cstdlib>
#include <cassert>
int add(int a, int b)
{
    return a+b;
}
```

run_simple_adder.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <cassert>
#include "simple_adder.h"
using namespace std;
int main()
{
    int a,b;
    cout<<"please enter two integers a and b: "<<endl;
    cin>>a>>b;
    cout<<"a+b="<<add(a,b)<<endl;
}

```

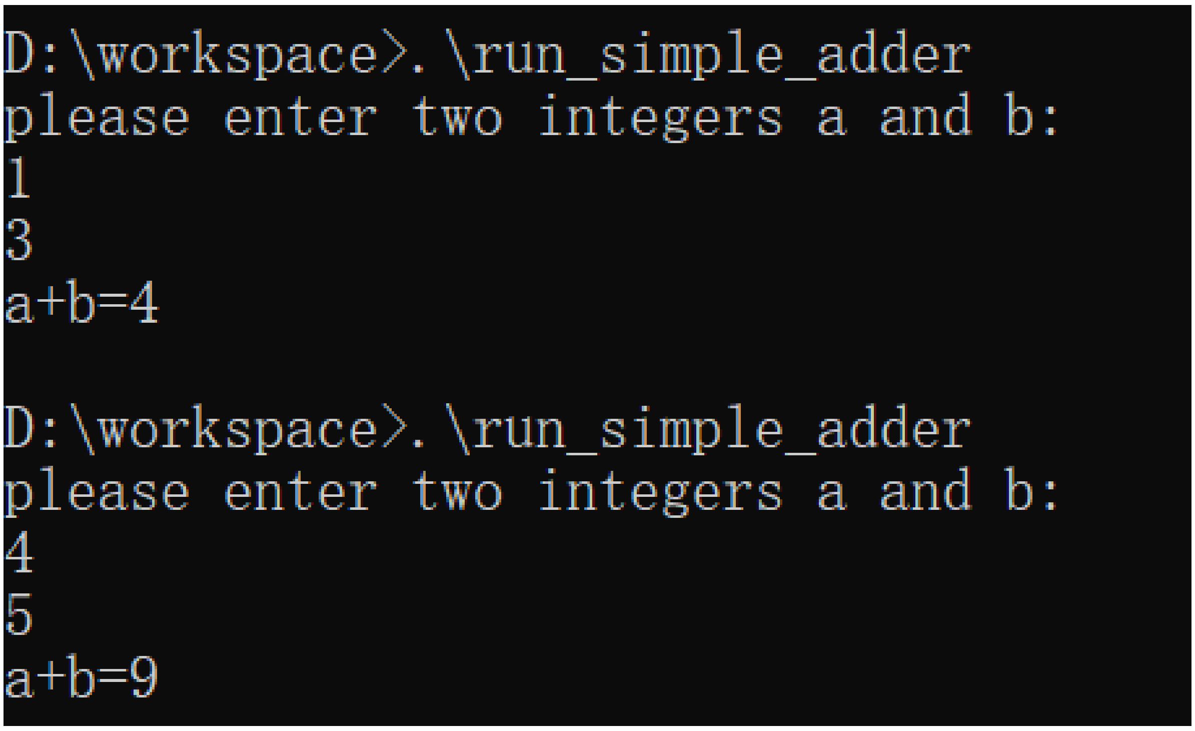
Makefile (without name extension)

```
all:run_simple_adder

run_simple_adder:run_simple_adder.o simple_adder.o
    g++ -o run_simple_adder run_simple_adder.o simple_adder.o

run_simple_adder.o:run_simple_adder.cpp
    g++ -c run_simple_adder.cpp

simple_adder.o:simple_adder.cpp
    g++ -c simple_adder.cpp

clean:
    rm-f run_simple_adder *.o
```



```
D:\workspace>make
g++ -c run_simple_adder.cpp
g++ -c simple_adder.cpp
g++ -o run_simple_adder run_simple_adder.o simple_adder.o
```

Now we have run_simple_adder.exe



```
D:\workspace>.\run_simple_adder
please enter two integers a and b:
1
3
a+b=4

D:\workspace>.\run_simple_adder
please enter two integers a and b:
4
5
a+b=9
```