南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Advanced Programming

Prof. Shiqi Yu (于仕琪)

yusq@sustech.edu.cn

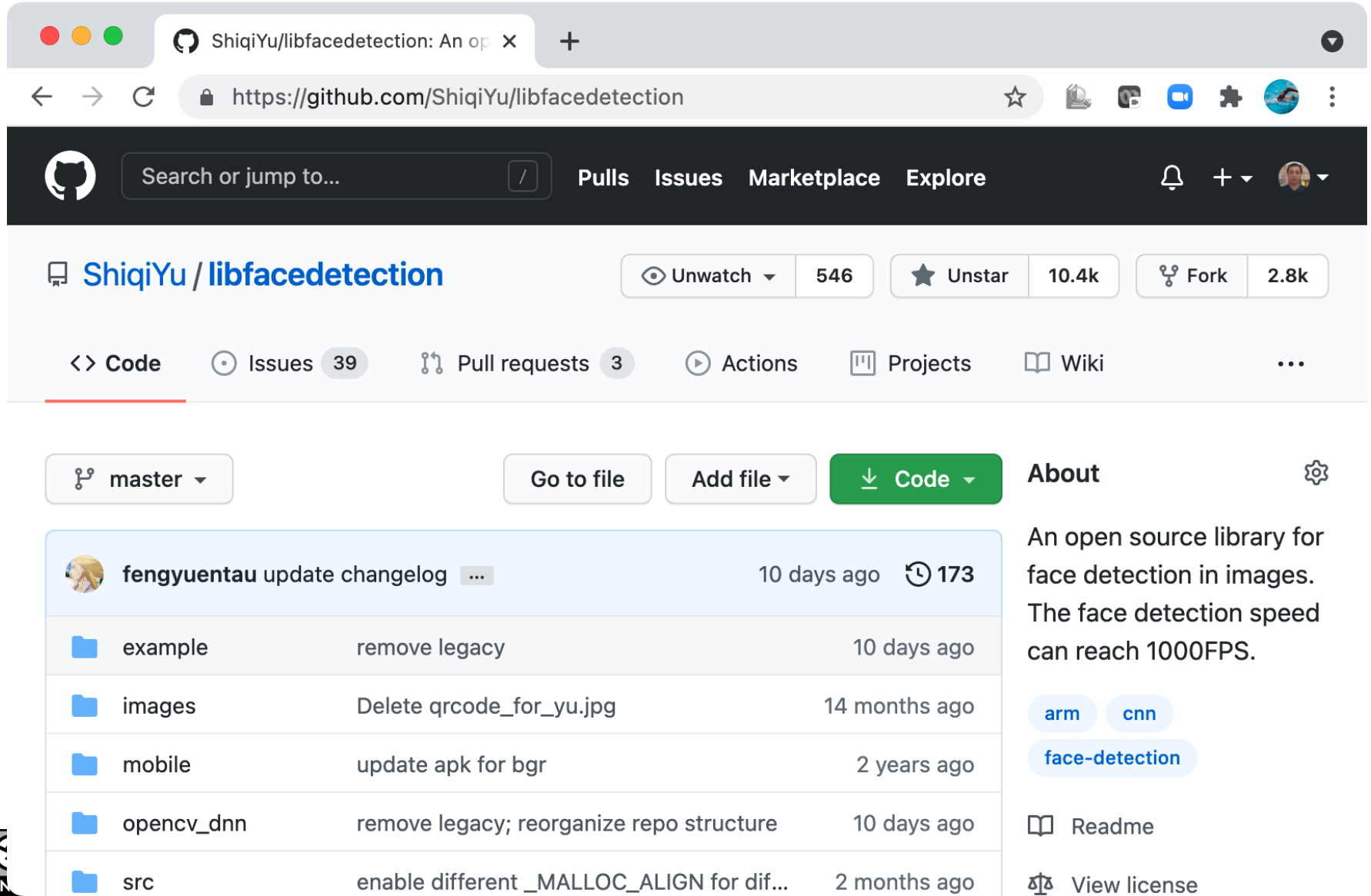
http://faculty.sustech.edu.cn/yusq/

About me

Prof. Shiqi Yu (于仕琪)

- Department of Computer Science and Engineering in Southern University of Science and Technology(南方科技大学计算机科学与工程系)
- **Office :** Room 312, South Tower, CoE Building
- **Email :** yusq@sustech.edu.cn
- Homepage: http://faculty.sustech.edu.cn/yusq/

My Open Source Project



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

About the Course

Grade Component

- Quiz: 5% (the best N-1 scores)
- Lab Attendance and Exercise: 5% (the best N-1 scores)
- Project: 65%
  - ~5 projects, some are easy projects
  - Grading standard:
    - 90-100: Finish all tasks almost perfectly
    - 80-90: Finish all tasks well
    - 70-80: Finish all tasks
- Exam: 25%

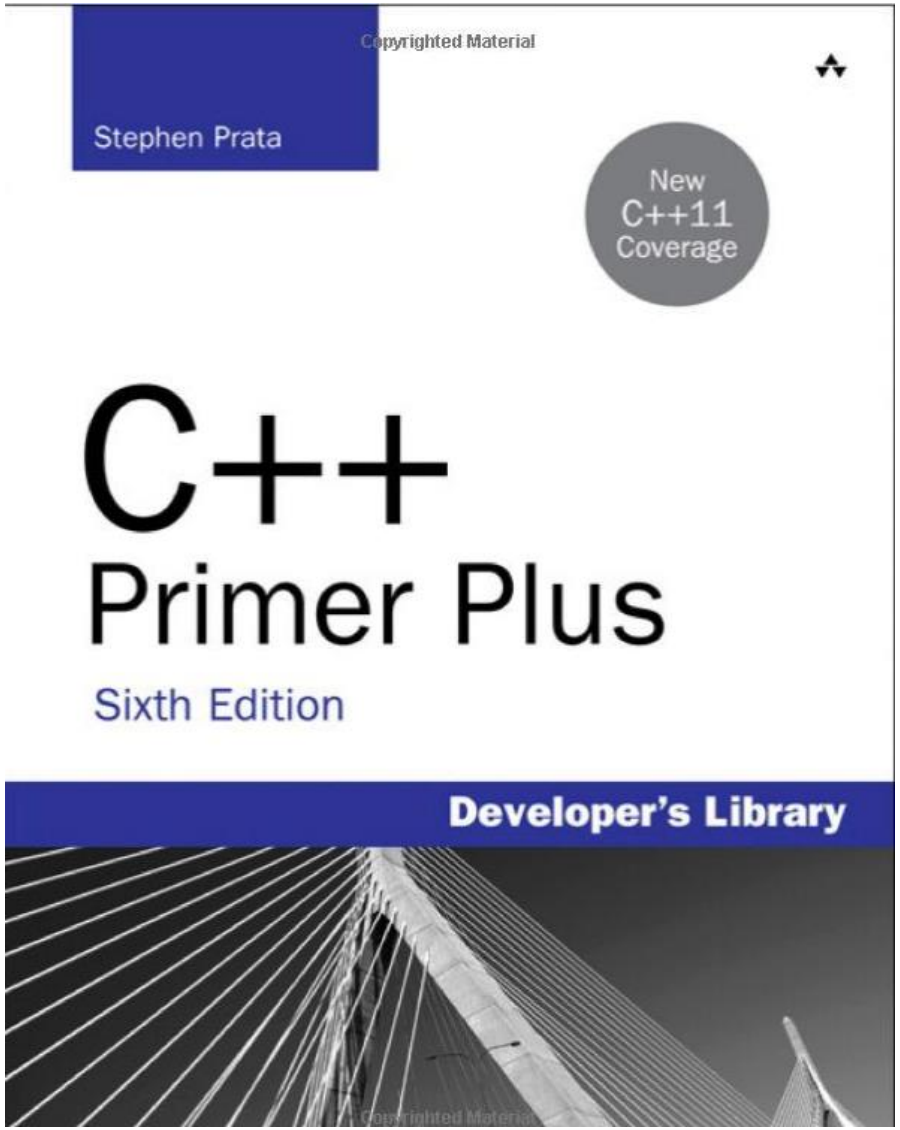
Honesty

- Get code from the internet for labs/assignments is perfectly **OK**
  - When you borrow, just say it.
  - You don't need to reinvent the wheel

- **DON'T** pretend that you are the author of something that you didn't write. Otherwise, the score will be **ZERO**!

Resources

- **Blackboard:**
  - C/C++ Program Design
- **Useful websites:**
  - https://en.cppreference.com/w/
  - https://www.w3schools.com/cpp/
  - http://cpp.sh/
  - https://www.onlinegdb.com/



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

The First Example

hello.cpp

```
//C++ example in C++11
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main()
{
   vector<string> msg {"Hello", "C++", "World", "!"};
   for (const string& word : msg)
   {
      cout << word << " ";
   }
   cout << endl;
}
```

Compile and run the program

- Compile hello.cpp

```
g++ hello.cpp
```

- Initialization of `msg` is a C++11 extension. We need

```
g++ hello.cpp --std=c++11
```

- Executable file can be generated as a.out. Change the output filename by -o option

```
g++ hello.cpp --std=c++11 -o hello
```

- Execute

```
./hello
```

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Different Programming Languages

Binary Instructions for CPU

- The instructions for CPU to run are binary.
  - 10110000 01100001
- Programming on punched tapes

https://en.wikipedia.org/wiki/Punched_tape

Assembly languages

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

               ************************************
               * FUNCTION: INITA - Initialize ACIA
               * INPUT: none
               * OUTPUT: none
               * CALLS: none
               * DESTROYS: acc A

0013           RESETA   EQU     %00010011
0011           CTLREG   EQU     %00010001

C003 86 13     INITA    LDA A  #RESETA   RESET ACIA
C005 B7 80 04           STA A  ACIA
C008 86 11              LDA A  #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A  ACIA

C00D 7E C0 F1           JMP    SIGNON    GO TO START OF MONITOR

               ************************************
               * FUNCTION: INCH - Input character
               * INPUT: none
               * OUTPUT: char in acc A
               * DESTROYS: acc A
               * CALLS: none
               * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A  ACIA      GET STATUS
C013 47                 ASR A            SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC    INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A  ACIA+1    GET CHAR
C019 84 7F              AND A  #$7F      MASK PARITY
C01B 7E C0 79           JMP    OUTCH     ECHO & RTS

               ************************************
               * FUNCTION: INHEX - INPUT HEX DIGIT
               * INPUT: none
               * OUTPUT: Digit in acc A
               * CALLS: INCH
               * DESTROYS: acc A
               * Returns to monitor if not HEX input

C01E 8D F0     INHEX    BSR    INCH      GET A CHAR
C020 81 30              CMP A  #'0       ZERO
C022 2B 11              BMI    HEXERR    NOT HEX
C024 81 39              CMP A  #'9       NINE
C026 2F 0A              BLE    HEXRTS    GOOD HEX
C028 81 41              CMP A  #'A
C02A 2B 09              BMI    HEXERR    NOT HEX
C02C 81 46              CMP A  #'F
C02E 2E 05              BGT    HEXERR
C030 80 07              SUB A  #7        FIX A-F
C032 84 0F     HEXRTS   AND A  #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF  HEXERR   JMP    CTRL      RETURN TO CONTROL LOOP
```

Assembly languages are more human readable

- 10110000 01100001

- MOV AL, 61h *; Load AL with 97 decimal (61 hex)*

https://en.wikipedia.org/wiki/Assembly_language

High Level Languages

- C: 1973
  - Developed by Dennis Ritchie and Ken Thompson at Bell Labs between 1969 and 1973.
- C++: 1979
  - Created by Bjarne Stroustrup as an extension of the C programming language
  - C with Classes
  - Renamed to C++
  - Now it's C++++++++++

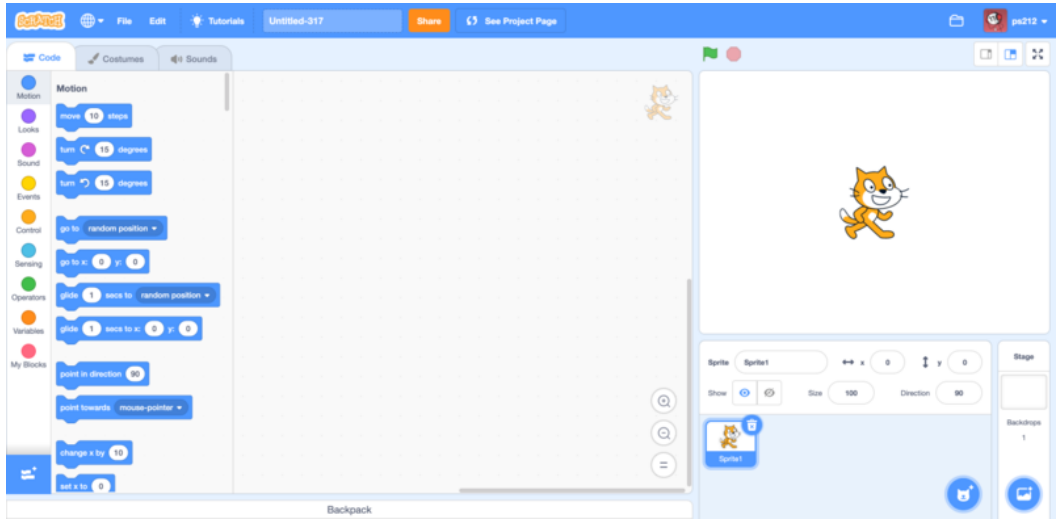
Higher Level Languages

- Java: 1995
  - I hate memory management in C/C++!
  - I want "Write once, run anywhere", not "write once, compile anywhere".
  - Grammar is similar with C++.
  - A Java compiler generates *.class files, not executable files.

- Python: 1990
  - I hate strict grammar!
  - I hate too many data types!

Even higher

- Scratch: 2002
  - ➢ I don’t like to type a keyboard



Something New

- Rust: 2012
  - Easier to manage memory than C/C++;
  - Better performance than Java;
  - Guarantee memory safety by static checking by the compiler.
  - Concise syntax, rich type system, and modern programming paradigms.

C and C++ are great! But...

- The grammar is complex, and **pointer ..**

C Language

C++

Year	C Standard[9]
1972	Birth
1978	K&R C
1989/1990	ANSI C and ISO C
1999	C99
2011	C11
2017	C17
TBD	C2x

Year	C++ Standard	Informal name
1998	ISO/IEC 14882:1998[29]	C++98
2003	ISO/IEC 14882:2003[30]	C++03
2011	ISO/IEC 14882:2011[31]	C++11, C++0x
2014	ISO/IEC 14882:2014[32]	C++14, C++1y
2017	ISO/IEC 14882:2017[33]	C++17, C++1z
2020	ISO/IEC 14882:2020[12]	C++20, C++2a

Advantages of C/C++

- Development language of most fundamental computer systems
  - Linux
  - MySQL
  - OpenCV
  - Backend of TensorFlow, PyTorch
  - ...
- High efficiency
  - Widely optimized compilers
  - Access memory directly
  - Excellent on computing
  - Important language for AI algorithm implementation

Similar languages

- C, C++, Java, and Rust

```c
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

```cpp
#include <iostream>
int main()
{
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

```java
public class Hello{
    public static void main(Str){
        System.out.println("Hello World!");
    }
}
```

```rust
fn main() {
    println!("Hello World!");
}
```

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Compile and Link


```
#include <iostream>

using namespace std;

int mul(int a, int b)
{
    return a * b;
}
int main()
{
    int a, b;
    int result;

    cout << "Pick two integers:";
    cin >> a;
    cin >> b;

    result = mul(a, b);

    cout << "The result is " << result << endl;
    return 0;
}
```

Two functions

- main(): called by startup code
- mul() is called in main()

Function prototypes and definitions

- function prototypes normally are put into head files (*.h; *.hpp)

```
int mul(int a, int b);
```

- function definitions normally are in source files (*.c; *.cpp)

```
int mul(int a, int b)
{
    return a * b;
}
```

Separate the source code into multiple files

main.cpp

```cpp
#include <iostream>
#include "mul.hpp"

using namespace std;
int main()
{
    int a, b;
    int result;

    cout << "Pick two integers:";
    cin >> a;
    cin >> b;

    result = mul(a, b);

    cout << "The result is " << result << endl;
    return 0;
}
```

mul.hpp

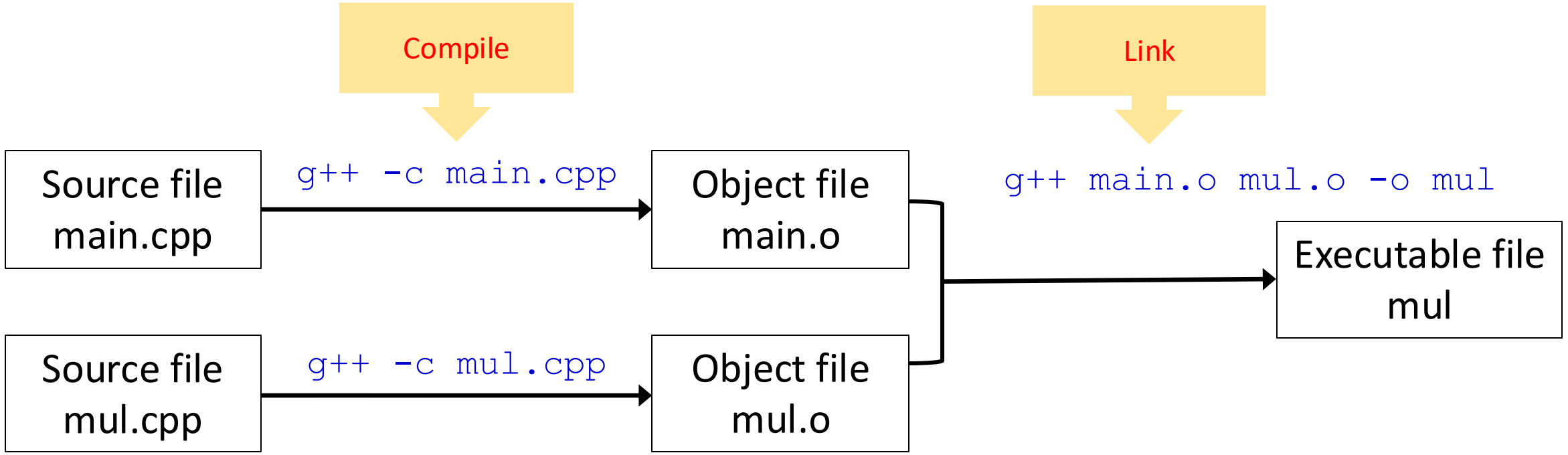
```cpp
#pragma once

int mul(int a, int b);
```

mul.cpp

```cpp
#include "mul.hpp"
int mul(int a, int b)
{
    return a * b;
}
```

Compile and link



Compilation errors

```
 9
10      cout << "Pick two integers:";
11      cin >> a;
12      cin >> b;
13
14      result = mul(a, b)
15
16      cout << "The result is " << result << endl;
17      return 0;
18  }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```
yushiqi@MacBook-Air-2 ch01 % g++ main.cpp -c
main.cpp:14:23: error: expected ';' after expression
    result = mul(a, b)
                      ^
                      ;
1 error generated.
```

- Normally caused by grammar error
- Please check the source code!

Link errors

- "Symbol not found”
- Function mul() is misspelled to Mul()

code > ch01 > mul.cpp > ...

```cpp
#include "mul.hpp"

int Mul(int a, int b)
{
    return a * b;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh

```
yushiqi@MacBook-Air-2 ch01 % g++ main.cpp -c
yushiqi@MacBook-Air-2 ch01 % g++ mul.cpp -c
yushiqi@MacBook-Air-2 ch01 % g++ main.o mul.o -o mul
Undefined symbols for architecture x86_64:
  "mul(int, int)", referenced from:
      main in main.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

Runtime errors

code > ch01 > mul.cpp > mul(int, int)

```
1   #include "mul.hpp"
2
3   int mul(int a, int b)
4   {
5       int c = a / b;
6       return a * b;
7   }
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TE

```
yushiqi@MacBook-Air-2    01 % ./mul
Pick two integers:2 0
zsh: floating point exception  ./mul
```

- The source code can be successfully compiled and linked.
- The floating point exception (divided by 0) will kill the program.
- It is a typical runtime error.

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Preprocessor and Macros

Preprocessor

- The preprocessor is executed before the compilation.
- Preprocessing directives begin with a # character
- Each directive occupies one line
- preprocessing instruction
  (define, undef, include, if, ifdef, ifndef, else, elif, endif, line, error, pragma)

```
#include <iostream>

#define PI 3.1415926535

#if defined(_OPENMP)
#include <omp.h>
#endif
```

`include` directive

Preprocess

Compile

main.cpp

```
#include "mul.hpp"
int main()
{
    //
}
```

```
int mul(int a, int b);
int main()
{
    //
}
```

main.o

Binary object file

Macros

Preprocess

Compile

circle.cpp

```
#define PI 3.14
double len(double r)
{
    return 2.0 * PI * r;
}
```

```
double len(double r)
{
    return 2.0 * 3.14 * r;
}
```

circle.o

Binary object file

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Simple Output and Input

C++ Style Output

- What is cout?

```
std::ostream cout;
```

- cout is an object of data type ostream in namespace std.

```
cout << "hello." << endl;
```

- << is an operator which is defined as follows

std::basic_ostream<CharT,Traits>::operator<<

```
basic_ostream& operator<<( short value );
basic_ostream& operator<<( unsigned short value );
basic_ostream& operator<<( int value );
basic_ostream& operator<<( unsigned int value );
```

- endl, an output-only I/O manipulator. It will output a new line character and flushes.

C++ Style Input

```
int a;
float b;
cin >> a;
cin >> b;
```

- Similarly, cin is an object of type std::istream.
- >> is an operator

C Style Output

```
int v = 100;
printf("Hello, value = %d\n", v);
```

- `int printf( const char *format, ... );` is a function
- `format`: a string specifying how to interpret the data
- %d will let the function interpret v as an integer

C Style Input

```
int v;
int ret = scanf("%d", &v);
```

- scanf reads data from stdin, and interpret the input as an integer and store it into v;

Why the examples have no GUI?

- The programs I used all have GUI. Why the examples have no GUI?
- GUI (graphical user interface) is not mandatory.
- GUI is for human beings to interact with computers.
- No all programs interact with human beings.
- We can also interact with the program in a command line window.
- We can call a GUI library to create a graphic window by many programming languages. Surely C/C++ can create a GUI window.

Command line arguments

- Do you still remember?

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

- g++ is an executable program/file
- There are three command line arguments

-
```
int main()
{
    /* ... */
}
```

-
```
int main(int argc, char *argv[])
{...
}
```

-
```
int main(int argc, char **argv)
{...
}
```

Command line arguments

argument.cpp

```
#include <iostream>

using namespace std;
int main(int argc, char * argv[])
{
    for (int i = 0; i < argc; i++)
        cout << i << ": " << argv[i] << endl;
}
```

```
yushiqi: ch01 $ ./argument mul.cpp -o main
0: ./argument
1: mul.cpp
2: -o
3: main
```

But

- I don’t like to compile a program in a command window
- IDE: Integrated development environment
  - Microsoft Visual Studio
  - Apple Xcode
  - Eclipes
  - Clion
  - …

- Visual Studio Code (VSCode) is an integrated development environment made by Microsoft for Windows, Linux and macOS