

# OOP Lab3

---

## C ++ Preprocessor

**Deadline:** 23:59. Saturday, May 21, 2016 (GMT+8)

For any further question, you can turn to Zhongyi Tong ([13302010039@fudan.edu.cn](mailto:13302010039@fudan.edu.cn)) or Jiacheng Xu ([13302010009@fudan.edu.cn](mailto:13302010009@fudan.edu.cn)) for help.

## Set up

---

In this lab, you are required to write a simple C++ preprocessor.

Download the starter code at:

```
ftp://10.132.141.33//classes/14/152 面向对象程序设计(陈辰)/LAB/lab3
```

You can find the following contents:

- `/test`: a test folder that contains raw C++ code segments
- `/test/run_tests.sh`: a grading script to test your processed code segment
- `lab3.cpp`: entrance for your preprocessor

**Note:** The package you download is a CLion project by default. If you are using other environments, adjust it as you wish. But remember to specify your environment in your documentation.

## Introduction

---

### What does the preprocessor do?

The preprocessor takes a look at your source code just before it goes off to the compiler, does a little formatting, and carries out any instructions you have given it.

#### Like what?

Well, preprocessor instructions are called *preprocessor directives*, and they all start with a `#`.

## Like #include?

Exactly. Each `#` command that the preprocessor encounters results in a modification to the source code in some way.

## Preprocessor directives

There are common directives like `#include`, `#define`, `#undef`, `#ifdef`, `#ifndef`, `#if`, `#endif`, etc. There are also predefined macros like `__FILE__`, `__TIME__`, etc.

In this lab, our preprocessor needs to deal with these directives in simple scenarios. There are also Preprocessor Operators, Predefined Macros, Pragas, or complex directives, which are left as bonus.

**Quiz question:** Describe the usage of each directive above and answer the question: why do we need a preprocessor to do this instead of using functions at runtime.

**References:** <http://www.cplusplus.com/doc/tutorial/preprocessor/>

## How to implement a preprocessor?

There are easy ways, and hard ways. To simplify, the preprocessor is just a string processor, that stores "tokens" and replaces them with regex in the following context. There are plenty of different angles, and how you abstract the problem is very important. Your preprocessor is supposed to have a well designed structure, with elegant code style.

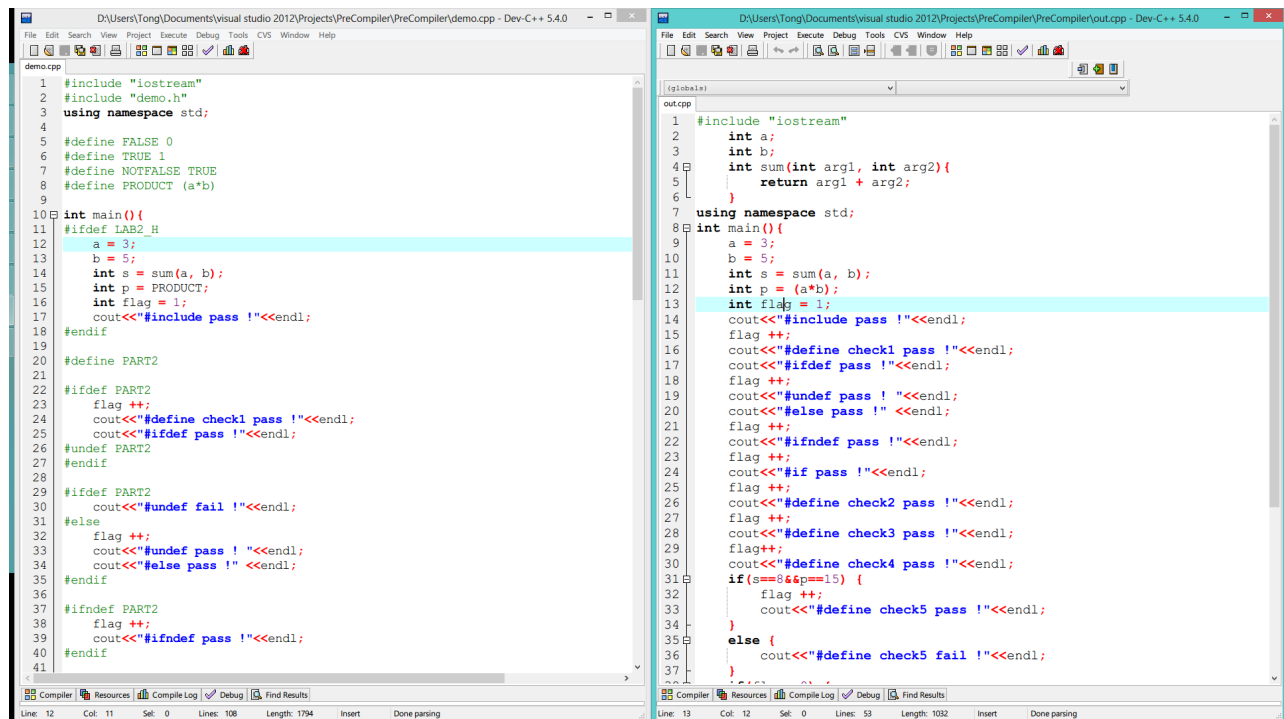
**References:** [The GNU C Preprocessor Internals](#), [MSDN C/C++ Preprocessor Reference](#), etc

## What to do in this lab?

- Implement your code. `lab3.cpp` provides an entrance for your preprocessor.
- Test your preprocessor. There are two test files in `/test` folder — `test1` is very simple and helps you debug at early stage, `test2` is complicated and used to verify cases in all scenarios.
- Run `test/run_tests.sh` to run all tests. Make sure you are under `/test`.
- If you are developing extra features, you must provide additional tests as well.
- Write well organized documentation to describe your project structure. Don't

forget to include the quiz question above.

**Left:** unprocessed code segment; **Right:** processed code segment.



```
1 #include "iostream"
2 #include "demo.h"
3 using namespace std;
4
5 #define FALSE 0
6 #define TRUE 1
7 #define NOTFALSE TRUE
8 #define PRODUCT (a*b)
9
10 int main(){
11 #ifdef LAB2_H
12     a = 3;
13     b = 5;
14     int s = sum(a, b);
15     int p = PRODUCT;
16     int flag = 1;
17     cout<<"#include pass !"<<endl;
18 #endif
19
20 #define PART2
21
22 #ifdef PART2
23     flag ++;
24     cout<<"#define check1 pass !"<<endl;
25     cout<<"#ifdef pass !"<<endl;
26 #undef PART2
27 #endif
28
29 #ifdef PART2
30     cout<<"#undef fail !"<<endl;
31 #else
32     flag ++;
33     cout<<"#undef pass !"<<endl;
34     cout<<"#else pass !"<<endl;
35 #endif
36
37 #ifndef PART2
38     flag ++;
39     cout<<"#ifndef pass !"<<endl;
40 #endif
41 }
```

```
1 #include "iostream"
2 int a;
3 int b;
4 int sum(int arg1, int arg2){
5     return arg1 + arg2;
6 }
7 using namespace std;
8 int main(){
9     a = 3;
10    b = 5;
11    int s = sum(a, b);
12    int p = (a*b);
13    int flag = 1;
14    cout<<"#include pass !"<<endl;
15    flag ++;
16    cout<<"#define check1 pass !"<<endl;
17    cout<<"#ifdef pass !"<<endl;
18    flag ++;
19    cout<<"#undef pass !"<<endl;
20    cout<<"#else pass !"<<endl;
21    flag ++;
22    cout<<"#ifndef pass !"<<endl;
23    flag ++;
24    cout<<"#if pass !"<<endl;
25    flag ++;
26    cout<<"#define check2 pass !"<<endl;
27    flag ++;
28    cout<<"#define check3 pass !"<<endl;
29    flag ++;
30    cout<<"#define check4 pass !"<<endl;
31    if(s==8&&p==15) {
32        flag ++;
33        cout<<"#define check5 pass !"<<endl;
34    }
35    else {
36        cout<<"#define check5 fail !"<<endl;
37    }
38 }
```

# Grading Criteria

## Directives (60%)

- `#include` - 10%
- `#define` basic (check1 to check5) - 10%
- `#ifdef` - 10%
- `#if` - 10%
- `#define` function (PART 3) - 10%
- `#define` function (PLUSES) - 10%
- bonus - up to 20%

You can get whole 60% if you pass two provided tests.

## Code (25%)

- Project structure - 20%
- Code style - 5%

## Documentation (15%)

- Quiz question - 5%

- Documentation - 10%

## Submission

---

Your submission should include the project directory with an `answer.txt`.

Compress your directory to `studentID_name.zip`.

Submit your work to:

```
ftp://10.132.141.33/classes/14/152 面向对象程序设计(陈  
辰)/WORK_UPLOAD/lab3
```