

Part VI

The C Preprocessor

[C and its Preprocessor](#)

[Hello World](#)

[Features](#)

[File Inclusion](#)

[Macro Substitution](#)

[Conditional Inclusion](#)

[Errors and Warnings](#)

[Phases](#)

C and its Preprocessor

A **preprocessor** is a program that **takes input data** (e.g., source files), **processes** (aka **preprocesses**) **it**, and moves its **output** to another program (e.g., a compiler).

Preprocessors Types

- **lexical:** replace character **tokens** by other **tokens** driven by user-defined rules
 - file inclusion
 - conditional inclusion
 - macro definition and substitution
- **syntactic:** transformation of **syntax trees** driven by user-defined rules
 - sometimes turing-complete computations
 - language extension

Hello World in C-Preprocessor

The **C preprocessor** is **lexical preprocessor** mostly used for the **C/C++ language**
Most simple beginners program outputting „Hello World“

C-Preprocessor
directives start with #

#warning Hello World

material/Chapter 02/00 Hello World/hello_world_preproc.c

- Run pre-processor only on that file (typically -E argument)

```
$ cc -E hello_world_preproc.c
# 1 "hello_world_preproc.c"
# 1 "<built-in>" 1
# 1 "<built-in>" 3
# 329 "<built-in>" 3
# 1 "<command line>" 1
# 1 "<built-in>" 2
# 1 "hello_world_preproc.c" 2
hello_world_preproc.c:1:2: warning: Hello World [-W#warnings]
#warning Hello World
^
```

1 warning generated.

C Preprocessor Features

FILE INCLUSION

MACRO SUBSTITUTION

CONDITIONAL
INCLUSION

STATE MESSAGES

COMPILER DIRECTIVES

File Inclusion

TOKEN REPLACEMENT
DIRECTIVE

```
#include "filename"
```

resp. #include <filename>

Macro Substitution (1)

Defined a macro:

```
#define identifier replacement
```

optional

Defined a macro
(function style):

```
#define identifier (identifier, ..., identifier) replacement
```

optional

optional

Check macro
definition exists:

```
defined (identifier)
```

optional

braces are optional

expands to 1 iff identifier is defined

Undefine a macro

```
#undef identifier
```

Macro Substitution (2)

Standard Pre-Defined Macros

- `__FILE__` expands to current **input file name**, e.g., `/home/marcus/arcade/main.c`
- `__LINE__` expands to current **input line number**, e.g., 23
- `__DATE__` expands to **date the preprocessor run**, e.g., Nov 06 2017
- `__TIME__` expands to **time the preprocessor run**, e.g., 21:49:32
- `__STDC__` expands to 1 if **compiler is ISO C conform**
- `__STD_VERSION__` expands to **C Standard version number**
- `__STD_HOSTED__` expands to 1 if **environment supports complete standard library**
- `__cplusplus` is defined iff a **C++ compiler is used**
- `__OBJC__` is defined and expands to 1 iff a **Objective-C compiler is used**
- `__ASSEMBLER__` is defined and expands to 1 iff **Assembler language is preprocessed**
- `__func__` expands to the current function name

Macro Substitution (3)

Common Pre-Defined Macros

Common pre-defined macros are implementation-defined

- macros to get version information of compiler
- checking whether strict ISO conformance is enforced
- macros to get various flags regarding enabled compiler optimization
- macros to get correct size in byte of various types
- macros to get the byte-order (little endian, big endian)
- ...

Macro Substitution (4)

User-specific Macros

Macros defined as argument to the preprocessor itself

- given as argument, e.g., `-D` flag in `cc` compiler
- existence and value check as for other macros
- user-defined semantic , e.g.,
 - platform-specific definition of symbols to check operating system
 - example `cc -DPLATFORM_UNIX -DPLATFORM_NAME=macOS`
 - fine-grained turning on or off of options
 - example `cc -DL0GGING=off -DBACKUP=on`

Conditional Inclusion

```
#if expression
```

Errors and Warnings

STATE MESSAGES

stop preprocessing and
output message as error

```
#error message
```

stop preprocessing and
output message as warning

```
#warning message
```

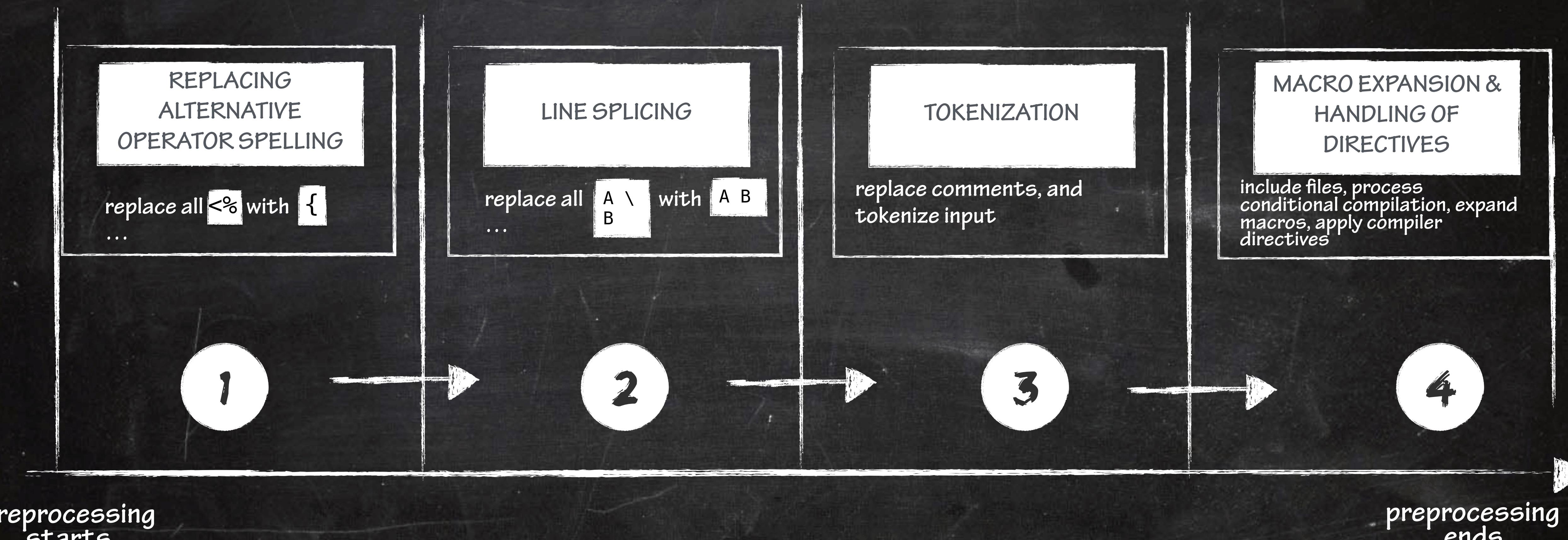
Errors and Warnings

COMPILER DIRECTIVES

```
#pragma text
```

C and its Preprocessor

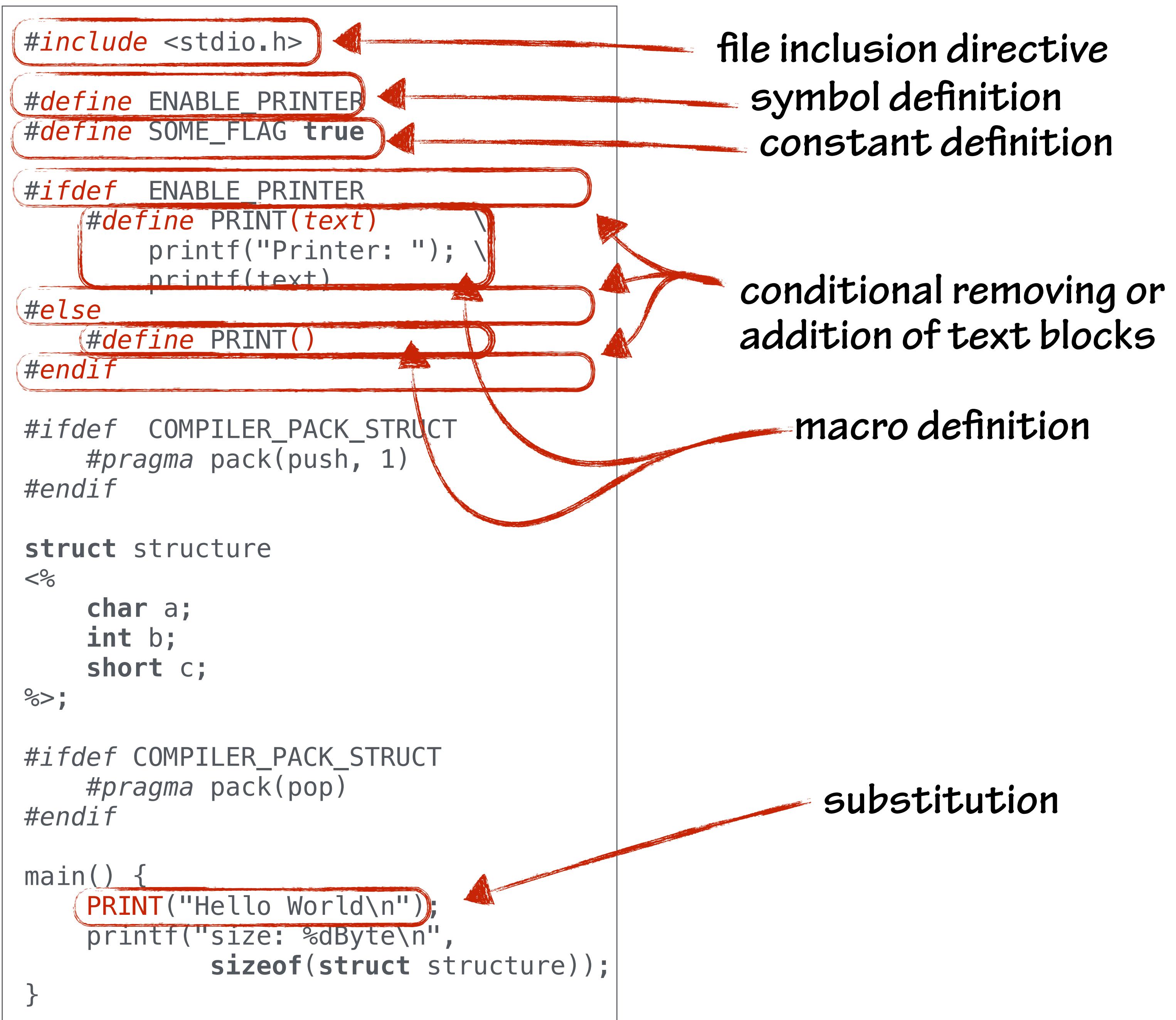
Phases



C and its Preprocess

Phase

0



C and its Preprocess

Phase

REPLACING
ALTERNATIVE
OPERATOR SPELLING

replace all <% with {
...}

1

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

#ifdef ENABLE_PRINTER
#define PRINT(text) \
    printf("Printer: "); \
    printf(text)
#else
#define PRINT()
#endif

#ifdef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif

struct structure
<%
char a;
int b;
short c;
%>;

#ifdef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

#ifdef ENABLE_PRINTER
#define PRINT(text) \
    printf("Printer: "); \
    printf(text)
#else
#define PRINT()
#endif

#ifdef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c;
};

#ifdef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

C and its Preprocess

Phase

LINE SPLICING

replace all  with 
...

2

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

#ifndef ENABLE_PRINTER
#define PRINT(text)
    printf("Printer: "); \
    printf(text)
#else
#define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c;
};

#ifndef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

```
#include <stdio.h>
```

```
#define ENABLE_PRINTER
#define SOME_FLAG
```

```
#ifndef ENABLE_PRINTER
#define PRINT(text) printf("Printer: "); \
else
#define PRINT()
#endif
```

```
#ifndef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif
```

```
struct structure
{
    char a;
    int b;
    short c;
};
```

```
#ifndef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif
```

```
main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

C and its Preprocess

Phase

TOKENIZATION

replace comments, and tokenize input

3

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

#ifndef ENABLE_PRINTER
    #define PRINT(text) printf..
#else
    #define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT
    #pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c
};

#ifndef COMPILER_PACK_STRUCT
    #pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

#ifndef ENABLE_PRINTER
    #define PRINT(text) printf..
#else
    #define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT
    #pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c
};

#ifndef COMPILER_PACK_STRUCT
    #pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

Legend:

- preprocessor tokens (red)
- don't know tokens (blue)

C and its Preprocessor

Phase

MACRO EXPANSION & HANDLING OF DIRECTIVES

include files, process conditional compilation, expand macros, apply compiler directives

4

```
#include <stdio.h>

#define ENABLE_PRINTER
#define SOME_FLAG

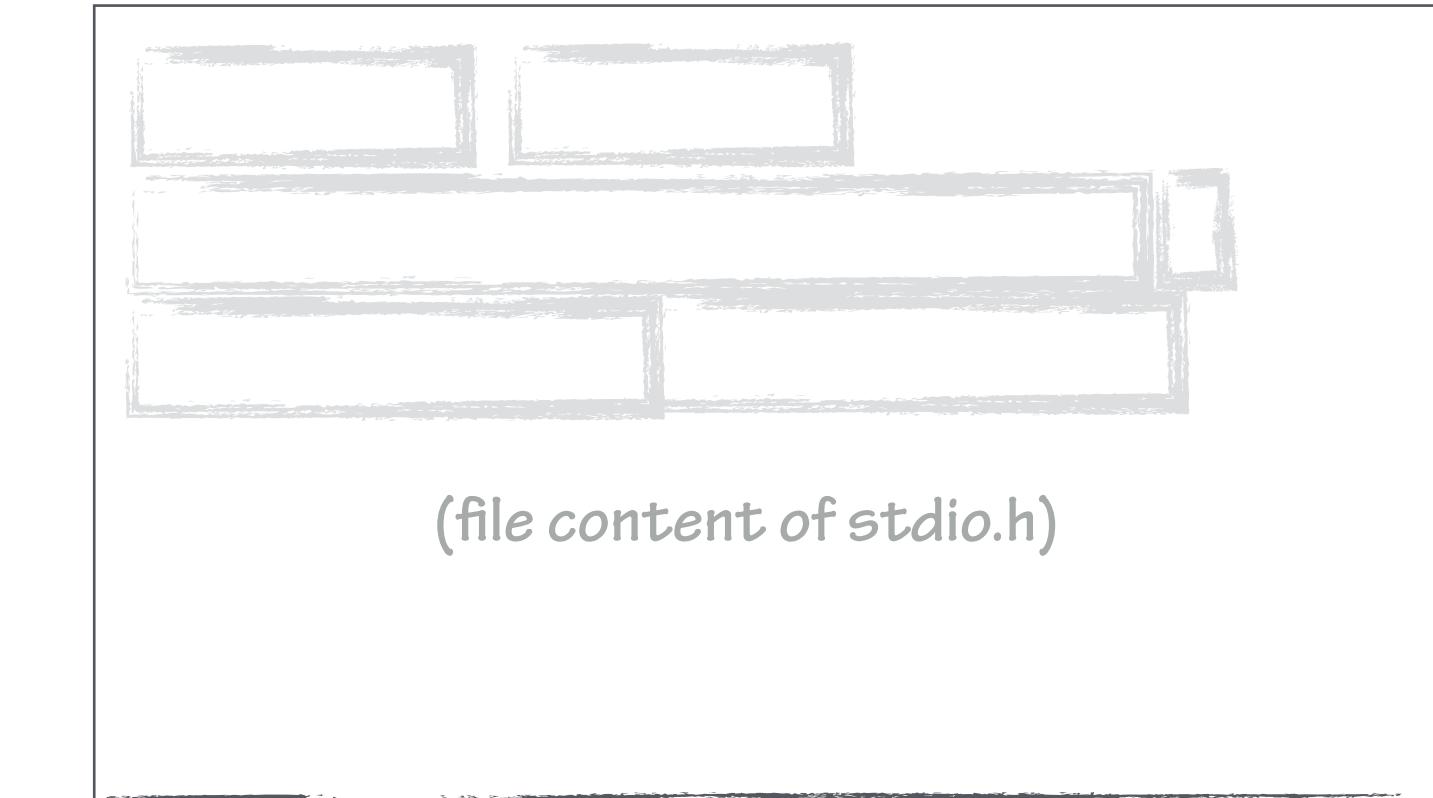
#ifndef ENABLE_PRINTER
#define PRINT(text) printf..
#else
#define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c;
};

#ifndef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```



```
#define ENABLE_PRINTER
#define SOME_FLAG

#ifndef ENABLE_PRINTER
#define PRINT(text) printf..
#else
#define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT
#pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c;
};

#ifndef COMPILER_PACK_STRUCT
#pragma pack(pop)
#endif
```

- **preprocessor tokens**
- **don't know tokens**

C and its Preprocess

Phase

MACRO EXPANSION &
HANDLING OF
DIRECTIVES

include files, process
conditional compilation, expand
macros, apply compiler
directives

4

known symbol unknown symbol

(file content of stdio.h)

```
#define ENABLE_PRINTER
#define SOME_FLAG ?

#ifndef ENABLE_PRINTER
    #define PRINT(text) printf...
#else
    #define PRINT()
#endif

#ifndef COMPILER_PACK_STRUCT ?
    #pragma pack(push, 1)
#endif

struct structure
{
    char a;
    int b;
    short c;
};

#ifndef COMPILER_PACK_STRUCT ?
    #pragma pack(pop)
#endif
```

material/Chapter 02/00 Hello World/cpp_phases.c

(file content of stdio.h)

```
#define PRINT(text) printf...

struct structure
{
    char a;
    int b;
    short c;
};

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure))
}
```

C and its Preprocess

Phase

MACRO EXPANSION &
HANDLING OF
DIRECTIVES

include files, process
conditional compilation, expand
macros, apply compiler
directives

4

(file content of stdio.h)

```
#define PRINT(text) printf("Printer: " text)

struct structure
{
    char a;
    int b;
    short c;
};

main() {
    PRINT("Hello World\n");
    printf("size: %dByte\n",
        sizeof(struct structure));
}
```

(file content of stdio.h)

```
struct structure
{
    char a;
    int b;
    short c;
};

main() {
    printf("Printer: ");printf("Hello W
    printf("size: %dByte\n",
        sizeof(struct structure))
}
```