



# **| Basic and Practice in Programming**

## Week 12



# File (1/4)

- File managing in C language
  - FILE data type (strictly, data struct in C library)
  - Three mode (Read/Write/Append)

```
struct FILE *fp;
```

- fopen
  - Function for opening a file with mode
  - Read(r) : open file as "Read-Only"
  - Write(w) : open file as "Write-Only"
  - Append(a) : open file as "Append-able"

```
fp = fopen(path, mode);
```

# File (2/4)

```
/* Practice 1 : File open - write */  
  
int main(void)  
{  
    FILE *fp;  
  
    fp = fopen("test", "w");  
  
    fprintf(fp, "Practice1");  
  
    fclose(fp);  
  
    return 0;  
}
```

Open a file "test" with Write-only mode

Close the file "test"

# File (3/4)

```
/* Practice 2 : File open - read*/  
  
int main(void)  
{  
    FILE *fp;  
    char str[20];  
  
    fp = fopen("test", "r");  
  
    fscanf(fp, "%s", str);  
    fprintf(stdout, "%s\n", str);  
  
    fclose(fp);  
  
    return 0;  
}
```

**Open a file "test" with Read-only mode**

**Close the file "test"**

# File (4/4)

```
/* Practice 3 : File open - append*/  
  
int main(void)  
{  
    FILE *fp;  
    char str[20];  
  
    fp = fopen("test", "a");  
  
    sprintf(str, "file open\n");  
    fprintf(fp, "%s\n", str);  
  
    fclose(fp);  
  
    return 0;  
}
```

Open a file "test" with Read-only mode

Close the file "test"

Mode can be passed with other mode  
fopen(path, "rw");

# Input/Output Operations (1/5)

- C language supports many functions for input/output
  - scanf/printf/
  - fscanf/fprintf
  - sscanf/sprintf
  - getchar/putchar
  - getc/putc
  - fgetc/fputc
  - ...

# Input/Output Operations (2/5)

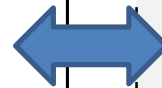
- Standard input/output
  - stdin : keyboard
  - stdout : monitor screen
  - stderr : monitor screen
- Scanf/printf as wrapper function

```
scanf("%d", &a) == fscanf(stdin, "%d", &a);
```

```
printf("%d", a) == fprintf(stdout, "%d", a);
```

# Input/Output Operations (3/5)

```
/* Practice 4 : fscanf/fprintf*/  
  
int main(void)  
{  
    int a;  
  
    fscanf(stdin, "%d", &a);  
    fprintf(stdout, "%d\n", a);  
  
    return 0;  
}
```



```
/* Practice 2 : File open - read*/  
  
int main(void)  
{  
    FILE *fp;  
    char str[20];  
  
    fp = fopen("test", "r");  
  
    fscanf(fp, "%s", str);  
    fprintf(stdout, "%s\n", str);  
  
    fclose(fp);  
  
    return 0;  
}
```

FILE is input/output stream from/to file



# Input/Output Operations (4/5)

```
/* Practice 5 : fgetc*/  
  
int main(void)  
{  
    char a;  
  
    a = fgetc(stdin);  
    fputc(a, stdout);  
    fputc('\n', stdout);  
  
    return 0;  
}
```

**fgetc(stdin) == getc(stdin) == getchar()**

**fputc(c, stdout) == putc(c, stdout) ==  
putchar(c)**

# Input/Output Operations (5/5)

- Make wrapper function
  - Using macro

```
/* Practice 6 : wrapper function*/
```

```
#define myprintf(fmt, str) fprintf(stdout, fmt, str)
```

```
int main(void)
```

```
{
```

```
    char *str = "wrapper function";
```

```
    myprintf("%s\n", str);
```

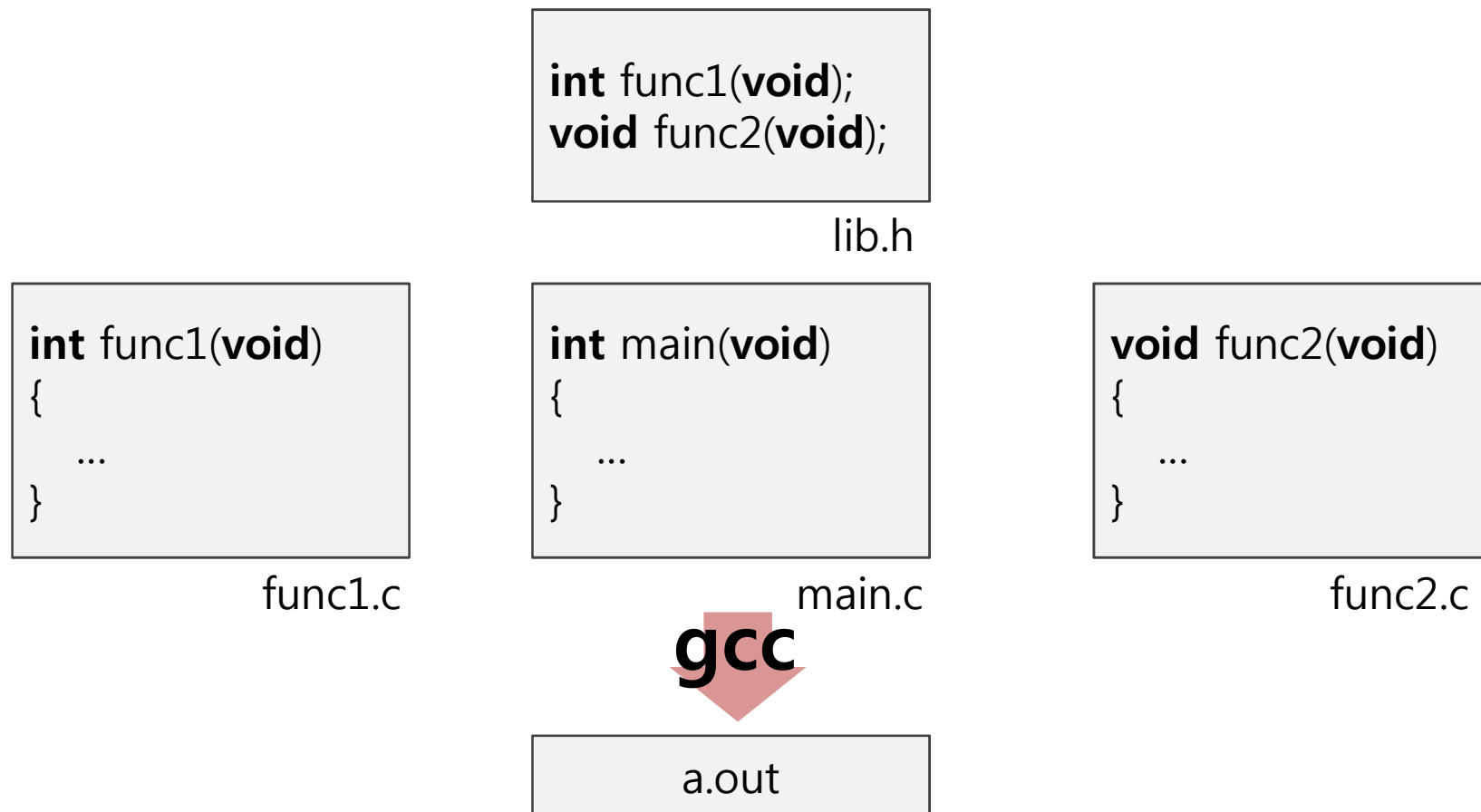
```
    return 0;
```

```
}
```

Handled by preprocessor

# Compile Many Files (1/2)

- Code can be written into many files



# Compile Many Files (2/2)

```
void func1(void);  
void func2(void);  
  
                                prac7_lib.h
```

```
/* practice 7 : Many files */  
#include "prac7_lib.h"  
  
int main(void)  
{  
    func1();  
    func2();  
  
    return 0;  
}  
  
                                prac7_main.c
```

```
#include <stdio.h>  
  
void func1(void)  
{  
    printf("func1!\n");  
}  
  
                                prac7_func1.c
```

```
#include <stdio.h>  
  
void func2(void)  
{  
    printf("func2!\n");  
}  
  
                                prac7_func2.c
```

```
$gcc prac7_main.c prac7_func1.c prac7_func2.c -o prac7
```

# Static Variable (revisit)

- Keyword "static"
  - Once initialize, never again

```
void func1(void);  
void func2(void);
```

```
/* practice 8 : static variable */  
int main(void)  
{  
    int i = 0;  
    for (i = 0; i < 10; i++) {  
        func1();  
        func2();  
    }  
    return 0;  
}
```

```
#include <stdio.h>  
  
void func 1(void)  
{  
    int a = 0;  
    printf("func1 %d\n", a);  
    a++;  
}
```

```
#include <stdio.h>  
  
void func 2(void)  
{  
    static int a = 0;  
    printf("func2 %d\n", a);  
    a++;  
}
```

# Extern Variable (1/3)

- C language supports method for using external variables
  - Keyword "extern"

```
/* practice 9 : extern variable 1*/  
  
int main(void)  
{  
    printf("%d\n", ext_val);  
}
```

prac9.c

```
int ext_val = 12;
```

prac9\_sub.c

```
$gcc prac9.c prac9_sub.c -o prac9
```

**Compile error occurred! Why?**

# Extern Variable (2/3)

- C language supports method for using external variables
  - Keyword "extern"

```
/* practice 10 : extern variable 2*/  
extern int ext_val;  
  
int main(void)  
{  
    printf("%d\n", ext_val);  
}
```

prac10.c

```
int ext_val = 12;
```

prac10\_sub.c

# Extern Variable (3/3)

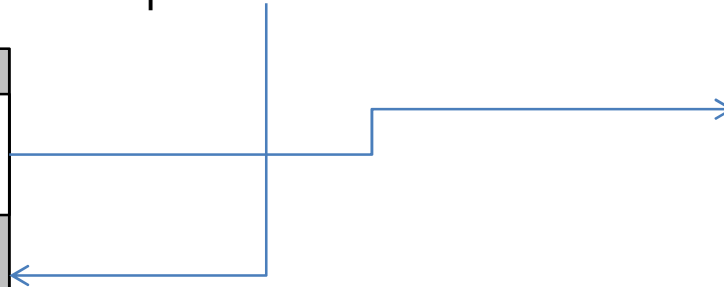
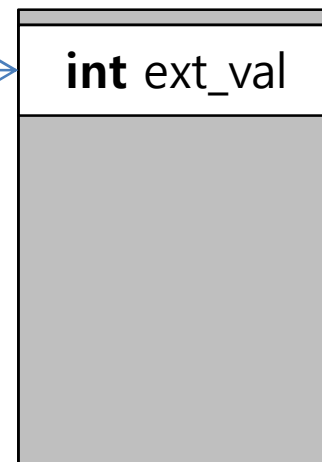
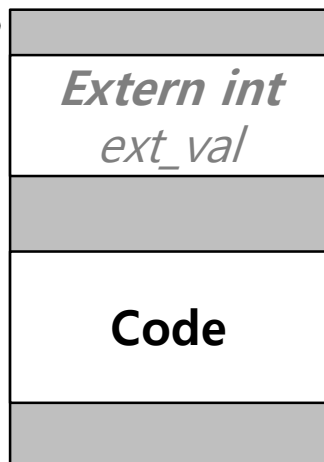
```
/* practice 10 : extern variable 2*/  
extern int ext_val;  
  
int main(void)  
{  
    printf("%d\n", ext_val);  
}
```

```
int ext_val = 12;
```

prac10\_sub.c

Empty slot

prac10.c





# Dipper in C

- Function pointer

```
/* practice 11 : function pointer*/  
  
void add (int a, int b)  
{  
    printf("add : %d\n", a + b);  
}  
  
void sub (int a, int b)  
{  
    printf("sub : %d\n", a - b);  
}
```

```
int main(void)  
{  
    void (*func) (int, int);  
  
    func = add;  
    func (3, 2);  
  
    func = sub;  
    func (3, 2);  
  
    return 0;  
}
```

# More Dipper in C

---

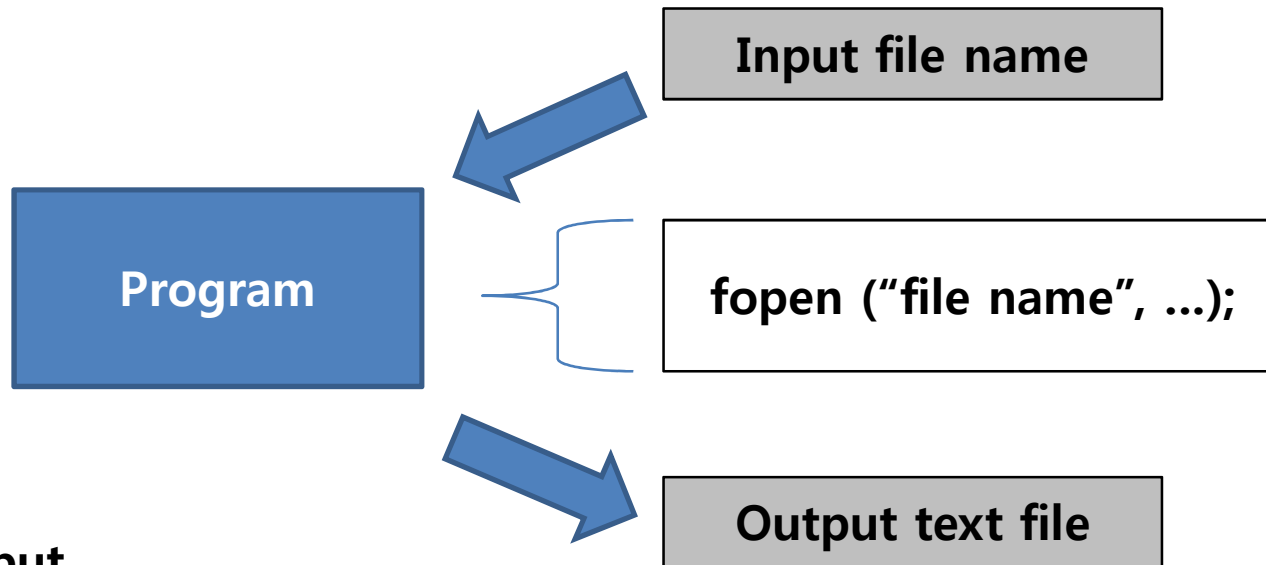
- Void \* (void pointer)
- Recursive call
- System call (fork/read/write...)
- Pre-processor

# Exercise

---

- White space remover (Due date : today's 11:59 pm)
  - Input : text file name
  - Output : White space removed text file
  - White space that includes tab, ' \n', ' '
    - You remove all of white space
- You use fopen function with given file name by redirection

# Exercise



## Sample input

```
./a.out < myheartwillgoon.txt
```

## myheartwillgoon.txt

```
Every night in my dreams  
I see you, I feel you  
That is how I know you go on
```

## Sample output

```
EverynightinmydreamsIseeyou,Ifeel  
youThatishowIknowyougoon
```