C Programming

Practice 12

Accessing Files

```c
#include <stdio.h>
int main(void)
{
    int sum = 0, val;
    FILE  *ifp;   /* infilepointer  */
    FILE  *ofp;  /* outfilepointer */
    ifp = fopen("in_file", "r");/* open for reading */
    ofp = fopen("out_file", "w");/* open for writing */
    …     /* do something with ifp and ofp*/
    fclose(ifp);
    fclose(ofp);
    return 0;
}
```

File access mode

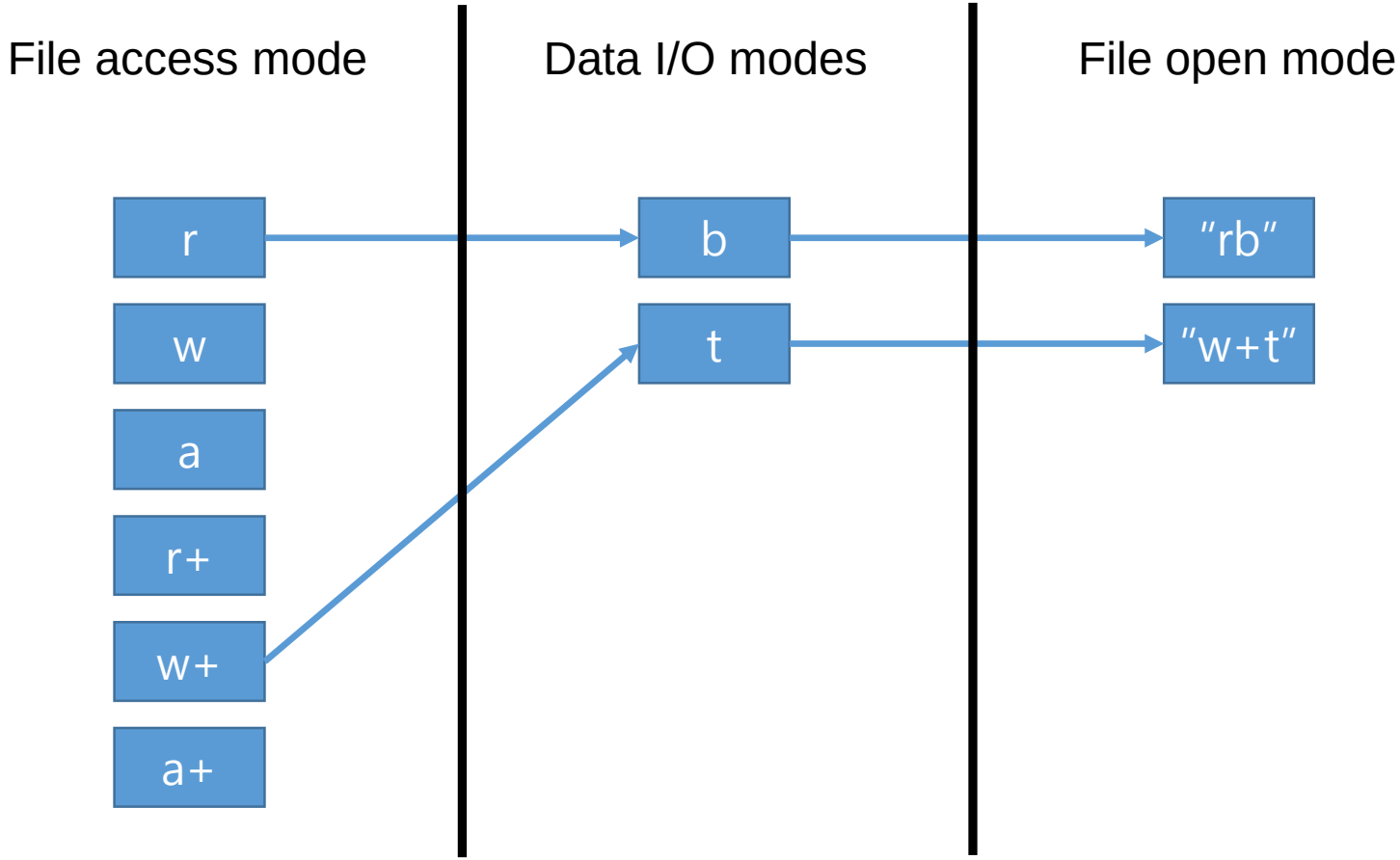
- Modes for opening files:
  - “r”: open text file for reading
  - “w”: open text file for writing – causes the file to be created if it does not exist and overwritten if it does.
  - “a”: open text file for appending
  - “r+”: open text file for reading and writing
  - “w+”: open text file for reading and writing
  - “a+”: open text file for reading and writing

- FILE* file = fopen(“database.txt”, “r”);
- fclose(file);

Data Input / Output modes

Data input and output modes	
Mode	Mean
t	Text mode
b	Binary mode

File open mode



File I/O functions

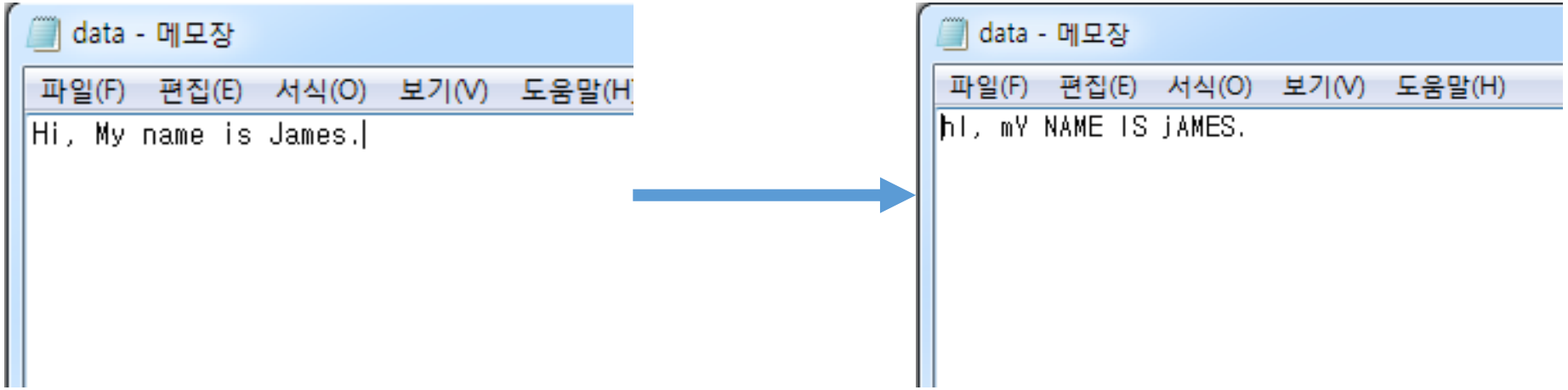
Standard input and output functions		
Character output	int putchar(int c)	int fputc(int c, FILE* stream)
Character input	int getchar(void)	Int fgetc(FILE* stream)
String output	int puts(const char* s)	int fputs(const char* s, FILE* stream)
String input	char* gets(char* s)	char* fgets(char* s, int n, FILE* stream)
Formatting output	int printf(const* format, …)	int fprintf(FILE* stream, const char* format, …)
Formatting input	int scanf(const char* format, …)	int fscanf(FILE* stream, const char* format, …)

File I/O functions – end of file

Standard input functions		Return value at the end of the file
Character input	Int fgetc(FILE* stream)	EOF(-1)
String input	char* fgets(char* s, int n, FILE* stream)	NULL pointer(0)
Formatting input	int fscanf(FILE* stream, const char* format, …)	EOF(-1)

Homework 23 – File I/O

- Read the string file (file name: data.txt)
- Save the output file(same file) after conversion to uppercase



malloc()

Syntax for malloc()
ptr = (cast type *)malloc(byte size);

```c
#include <stdio.h>
#include <stdlib.h>
 int main()
{
    int *pi;
    pi = (int*)malloc(sizeof(int));
    *pi = 3;
    printf("%d₩n", *pi);
    free(pi);
    return 0;
}
```

calloc()

Syntax for calloc()

ptr = (cast type *)calloc(byte size);

```
#include <stdio.h>
#include <stdlib.h>
 void main()
{
    int i, *mall, *call;
    mall=(int *)malloc(3 * sizeof(int)); //dynamic allocation using malloc
    call=(int *)calloc(3, sizeof(int));      //dynamic allocation using calloc
    printf("malloc : ");
    for(i=0; i<3; i++){
        printf("%d ", mall[i]);
    }
    printf("₩ncalloc : ");
    for(i=0; i<3; i++){
        printf("%d ", call[i]);
    }
    putchar('₩n');
}
```

realloc()

Syntax for realloc()
ptr = realloc(ptr, new size);

```c
#include <stdio.h>
#include <stdlib.h>
 void main()
{
    int i, *mall;
    mall=(int *)malloc(3 * sizeof(int)); //dynamic allocation using malloc
    printf("input a num: ");
    scanf("%d", &i);
    mall=(int *)realloc(mall, i*sizeof(int)); //dynamic allocation using realloc
}
```

free()

Syntax for free()
free(ptr);

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
    int *pi;
    pi = (int*)malloc(sizeof(int));
    *pi = 3;
    printf("%d₩n", *pi);
    free(pi);
    return 0;
}
```

Homework 24 – dynamic allocation

- Input a number from a user
- The user can continue input a number until input ‘-1’
- If -1 is entered, the program must show entered numbers

```
Number ? 1
Number ? 6
Number ? 5
Number ? 3
Number ? 4
Number ? 7
Number ? 8
Number ? 9
Number ? 10
Number ? -1
1, 6, 5, 3, 4, 7, 8, 9, 10,
```

C Programming

Practice Summary

While statement structure

```
int i = 0; // variable for while statement.

while( i < 10 ) // i < 10 is a conditional sentence.
{
        printf("%d",i); // print the i value.
        i = i + 1; // change the i value.
}
```

Do-While statement structure

```
do{

    1. statements
    2. Increment or Decrement statement

} while ( condition statement  );
```

switch & if-else

```
int num;
scanf ( "%d", &num );

switch (num) {
case 0:
   printf ( "ZERO₩n" );
   break;
case 1:
   printf ( "ONE₩n" );
   break;
default:
   printf ( "None of them₩n" );
   break;
}
```

```
int num;
scanf ( "%d", &num );

if (num==0)
   printf ( "ZERO₩n" );
else if (num==1)
   printf ( "ONE₩n" );
else
   printf ( "None of them₩n" );
```

For statement

```
for(i=0; i<10; i++) {
```

Counter Initialization

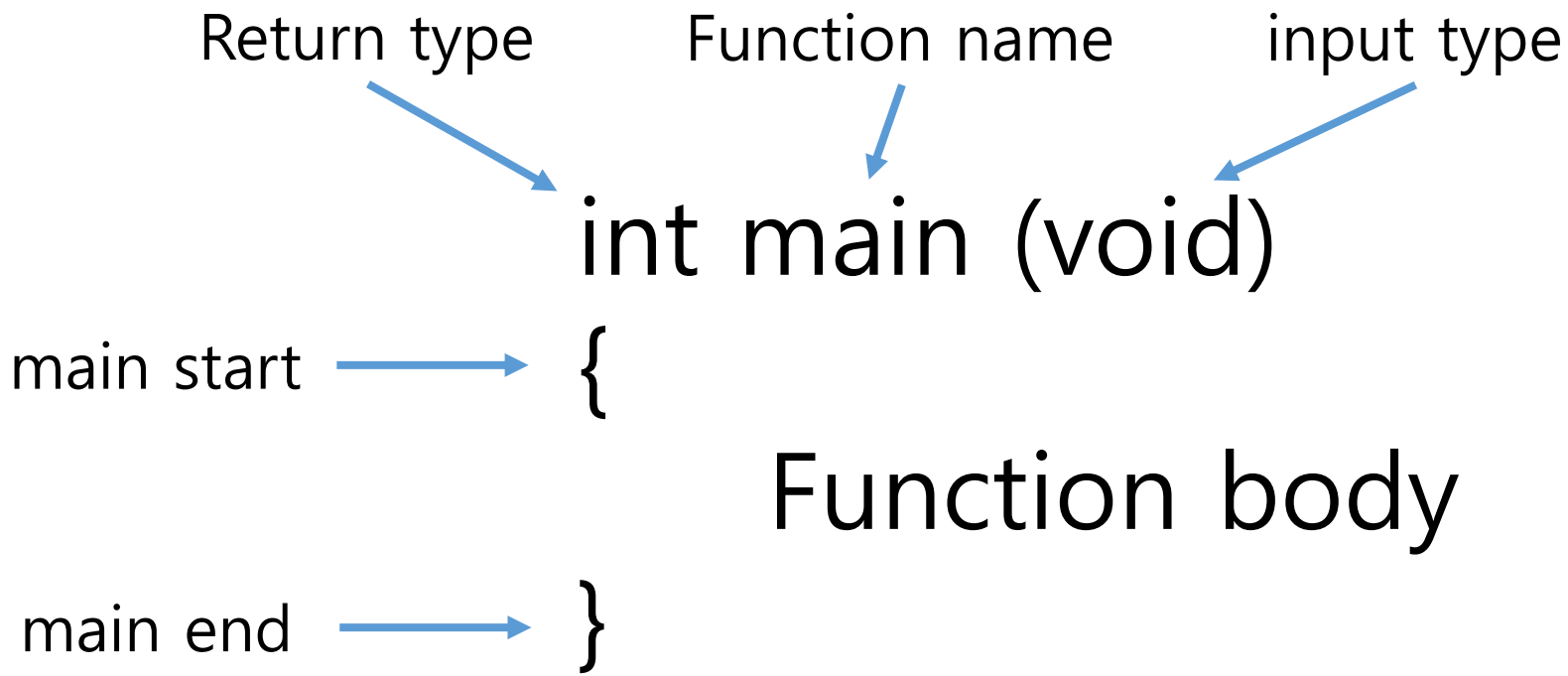
Repeat conditions

Counter Change: ex) ++i, i+2, --i, i--, i+2 등

statement

```
}
```

Function structure



Function structure

```c
#include <stdio.h>
void prn_message(void); /* function prototype*/
int main(void)
{
    prn_message();          /* function invocation*/
    return 0;
}
/*function definition*/
void prn_message(void)  /* function header*/
                        /* function body*/
{
    printf("Howdy!₩n");
}
```

Array

- Define array

```
int grade [3];
int grade [3] = {0};
char characters[3];
```


```
int a[] = {3, 4, 5};
```

The structure Type

```
struct Man { // Different data type !
    char name[30];
    int student_num[13];
    int tel[20];
    char addr[50];
};

struct Man member = { ... };
```

Linked-list structure

```
#include <stdio.h>  /* NULL is defined here */
#include <stdlib.h>
typedef char  DATA; /* use char in examples */
struct linked_list{
    DATA                d;
    struct linked_list *next;
};
typedef struct linked_list  ELEMENT;
typedef ELEMENT *           LINK;
```

Homework form

- Homework submission e-mail:

hizorro99@naver.com

- E-mail title: day(Thursday or Friday)_name_#week
  - Ex) Friday_james_week14
  - Ex) 목요일반_장원철_14주차
- File title: student id_name_#.c
  - Ex) 2014123456_james_23.c (or .cpp)
  - Ex) 2014123456_james_24.c (or .cpp)