Lecture 12. 파일 입출력

BAE SEKANG(배세강)

Contents

File I/O

♣ Standard I/O

- 파일 입출력: 파일을 통해 데이터를 입력 받거나 출력하는 방식
- scanf(): 키보드에서 입력, printf(): 화면에 출력

- standard I/O: 표준 입력과 표준 출력을 사용하는 입출력 방식

  stdin: 기본 입력 통로; keyboard
  stdout: 기본 출력 통로; monitor
  stderr: 오류 출력 통로; monitor

- stdio.h: 표준 입출력을 사용하기 위한 헤더 파일

File I/O

♣ Text File

- Text File: 사람이 읽을 수 있는 문자 형태로 저장된 파일
  ex) .txt, .csv, .c, .html,…
- pros: 사람이 읽기 쉽다, 다른 프로그램과 데이터를 주고 받기 쉽다
- cons: 파일의 크기가 크다, 읽을 때 다시 숫자로 변환해야 한다

File I/O

♣ Binary File

- Binary File: 이진 형식의 데이터로 저장된 파일
  ex) .bin, .jpg, .dat, .exe, …
- pros: 많은 데이터를 빠르게 저장하고 읽을 수 있다
- cons: 사람이 읽기 어렵다, OS나 시스템 환경에 따라 해석이 다를 수 있다

0	0011 0000	L	0100 1100	g	0110 0111	"	0010 0010
1	0011 0001	M	0100 1101	h	0110 1000	'	0010 0111
2	0011 0010	N	0100 1110	i	0110 1001	(	0010 1000
3	0011 0011	O	0100 1111	j	0110 1010	)	0010 1001
4	0011 0100	P	0101 0000	k	0110 1011	*	0010 1010
5	0011 0101	Q	0101 0001	l	0110 1100	+	0010 1011
6	0011 0110	R	0101 0010	m	0110 1101	,	0010 1100
7	0011 0111	S	0101 0011	n	0110 1110	-	0010 1101
8	0011 1000	T	0101 0100	o	0110 1111	.	0010 1110
9	0011 1001	U	0101 0101	p	0111 0000	/	0010 1111
		V	0101 0110	q	0111 0001	:	0011 1010
A	0100 0001	W	0101 0111	r	0111 0010	;	0011 1011
B	0100 0010	X	0101 1000	s	0111 0011	=	0011 1101
C	0100 0011	Y	0101 1001	t	0111 0100	?	0011 1111
D	0100 0100	Z	0101 1010	u	0111 0101		
E	0100 0101			v	0111 0110		
F	0100 0110	a	0110 0001	w	0111 0111		
G	0100 0111	b	0110 0010	x	0111 1000		
H	0100 1000	c	0110 0011	y	0111 1001		
I	0100 1001	d	0110 0100	z	0111 1010		
J	0100 1010	e	0110 0101				
K	0100 1011	f	0110 0110	!	0010 0001		

File I/O

♣ FILE Pointer

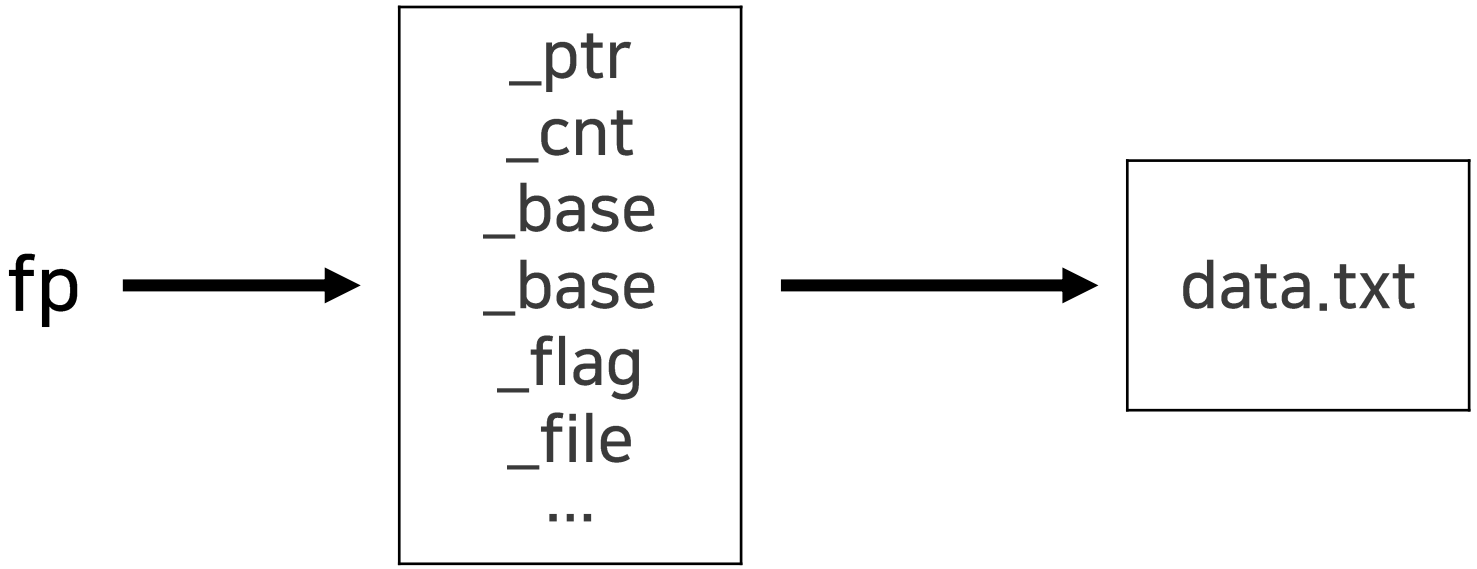
- C에서 파일을 다룰 때 사용하는 포인터
- FILE: stdio.h에 정의되어 있는 자료형
- 파일 포인터가 파일 자체는 아님에 주의

```
#include <stdio.h>

// fp는 파일에 접근하기 위한 정보를 가리키는 포인터
FILE *fp;
```

File I/O

♣ FILE Pointer

```
struct _iobuf {
    char *_ptr;
    int   _cnt;
    char *_base;
    int   _flag;
    int   _file;
    int   _charbuf;
    int   _bufsiz;
    char *_tmpfname;
};

typedef struct _iobuf FILE;
```



File Open & Close

♣ fopen()

- 파일을 여는 함수
- FILE *fopen(const char *filename, const char *mode);

```
FILE *fp;
fp = fopen("data.txt", "r"); // r은 read mode

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}

/* read or write data */

fclose(fp);
```

File Open & Close

♣ File Mode

Mode	Meaning	Explanation
“r”	read	읽기 모드
“w”	write	쓰기 모드, 기존 내용 삭제
“a”	append	추가 모드, 파일 끝에 내용 추가
“r+”	read/write	읽기와 쓰기
“w+”	read/write	읽기와 쓰기, 기존 내용 삭제
“a+”	read/append	읽기와 추가

- Binary File I/O를 위한 mode는 ‘b’를 붙인다. i.e.) rb, wb, ab, rb+, wb+, ab+

File Open & Close

♣ Example: “r” mode

- 이미 존재하는 파일을 읽을 때 사용
- 파일이 없으면 새로 만들지 않고 실패(NULL 반환)

```
FILE *fp = fopen("data.txt", "r");

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}
char ch;
while (fscanf(fp, "%c", &ch) == 1) {
    printf("%c", ch);
}

fclose(fp);
```

File Open & Close

♣ Example: “w” mode

- 파일이 없으면 새로 만들어 씀
- 파일이 이미 존재하면 **기존 내용을 모두 삭제**하고 새로 씀

```
FILE *fp = fopen("result.txt", "w");

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}

fprintf(fp, "Hello File I/O₩n");

fclose(fp);
```

File Open & Close

♣ Example: “a” mode

- 파일 끝에 데이터를 추가하기 위해 사용(기존 내용을 삭제하지 않음)
- 파일이 없다면 새로 만든다.

```
FILE *fp = fopen("log.txt", "a");

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}

fprintf(fp, "New log message₩n");

fclose(fp);
```

File Open & Close

♣ NULL check

- fopen()은 파일 열기에 실패하면 NULL을 반환

```
FILE *fp = fopen("data.txt", "r");

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}
```

[파일 열기에 실패하는 경우]

- 읽으려는 파일이 존재하지 않는 경우
- 파일 이름/경로가 잘못된 경우
- 파일 접근 권한이 없는 경우
- 디스크 문제 또는 기타 오류가 발생한 경우

File Open & Close

♣ fclose()

- 파일 사용이 끝나면 반드시 fclose()를 호출해 파일을 닫아야 한다.

```
FILE *fp = fopen("data.txt", "r");

if (fp == NULL) {
    printf("File open failed₩n");
    return 1;
}

/* use file */

fclose(fp);
```

- 파일을 닫은 후, 더 이상 해당 FILE 포인터로 입출력을 하면 안된다.

File Open & Close

♣ Example: fclose()

```
#include <stdio.h>

int main(void) {
    FILE *fp;

    fp = fopen("output.txt", "w");
    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }
    fprintf(fp, "Hello, File I/O!₩n");
    fprintf(fp, "This is a text file.₩n");

    fclose(fp);
    return 0;
}
```

Text File I/O

♣ fprintf()

- 파일에 형식화된 데이터를 출력하는 함수
- fprintf(FILE *fp, const char *format, ...);

```
FILE *fp = fopen("student.txt", "w");

fprintf(fp, "%s %d₩n", "Kim", 90);
fprintf(fp, "%s %d₩n", "Lee", 85);

fclose(fp);
```

```
Kim 90
Lee 85
```

student.txt

- fprintf(stdout, "%s %d₩n", "Kim", 90);와 다른 점은?

Text File I/O

♣ Example: fprintf()

```
#include <stdio.h>

int main(void) {
   FILE *fp;
   fp = fopen("student.txt", "w");

   if (fp == NULL) {
      printf("File open failed₩n");
      return 1;
   }
   fprintf(fp, "%s %d₩n", "Kim", 90);
   fprintf(fp, "%s %d₩n", "Lee", 85);
   fprintf(fp, "%s %d₩n", "Park", 95);

   fclose(fp);
   return 0;
}
```

Text File I/O

[Practice]

현재 경로에 practice1.txt를 생성 후 “Hello, C Language!”의 내용을 갖도록 하는 프로그램 코드를 작성한다.


```
Hello, C Language!
```

Text File I/O

[Practice] - Answer

```
#include <stdio.h>

int main(void) {
    FILE *fp;

    fp = fopen("practice1.txt", "w");

    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }

    fprintf(fp, "Hello, C Language!");

    fclose(fp);

    return 0;
}
```

Text File I/O

♣ fscanf()

- 파일에서 형식화된 데이터를 읽는 함수
- fscanf(FILE *fp, const char *format, ...);

```
char name[20];
int score;

// 파일에서 Kim과 90을 읽어 각각 name과 score에 저장한다.
fscanf(fp, "%19s %d", name, &score);

printf("%s %d₩n", name, score);
```

```
Kim 90
Lee 85
```

student.txt

Text File I/O

♣ Example: fscanf()

```
#include <stdio.h>

int main(void) {
   FILE *fp;
   char name[20];
   int score;
   fp = fopen("student.txt", "r");
   if (fp == NULL) {
      printf("File open failed₩n");
      return 1;
   }

   fscanf(fp, "%s %d", name, &score);
   printf("%s %d₩n", name, score);
   fscanf(fp, "%s %d", name, &score);
   printf("%s %d₩n", name, score);
   fscanf(fp, "%s %d", name, &score);
   printf("%s %d₩n", name, score);

   fclose(fp);
   return 0;
}
```

Text File I/O

♣ fputs() / fgets()

- fputs(): 문자열을 파일에 출력하는 함수
- fgets(): 파일에서 문자열 또는 한 줄을 읽는 함수
- fputs("Hello₩n", fp);
- fgets(str, sizeof(str), fp);

```
char line[100];

fgets(line, sizeof(line), fp);
printf("%s", line);

// fgets()는 줄바꿈 문자('₩n')까지 함께 읽는다.
```

Text File I/O

♣ Example: fgets()

```
#include <stdio.h>

int main(void) {
    FILE *fp;
    char line[100];
    fp = fopen("student.txt", "r");

    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }

    while (fgets(line, sizeof(line), fp) != NULL) {
        printf("%s", line);
    }

    fclose(fp);
    return 0;
}
```

Text File I/O

♣ fgetc() / fputc()

- 파일에서 문자 하나를 읽는/쓰는 함수
- int ch = fgetc(fp); // fgetc의 반환형은 char가 아닌 int임에 유의
- fputc('A', fp);

```
int ch;

while ((ch = fgetc(fp)) != EOF) {
    putchar(ch);
}
```

- EOF(End Of File): 파일의 끝을 나타내는 값

Text File I/O

♣ EOF(End Of File)

- 파일의 끝 또는 입력 종료를 나타내는 특별한 값
- stdio.h에 정의된 symbolic constant; 일반적으로 #define EOF (-1)
- 실제 파일 끝에 EOF 값이 들어 있는 것은 아니다.
  ➔ 파일을 읽다가 더 이상 읽을 데이터가 없다면 C의 입력 함수가 EOF를 반환한다.

```
int ch;

// 파일의 끝까지 도달할 때까지 문자 하나씩 출력한다.
while ((ch = fgetc(fp)) != EOF) {
    putchar(ch);
}
```

Text File I/O

[Practice]

현재 경로에 students.txt 파일을 생성하고, 다음 [내용]을 저장한다. 이후 [내용]을 읽어 화면에 세 학생 점수의 평균 점수를 출력한다.

[내용]
Kim 90
Lee 85
Park 95

[출력]
90.00

Text File I/O

[Practice] - Answer

```
#include <stdio.h>

int main(void) {
    FILE *fp;
    char name[20];
    int score, sum = 0, count = 0;
    fp = fopen("students.txt", "w");
    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }

    fprintf(fp, "Kim 90₩n");
    fprintf(fp, "Lee 85₩n");
    fprintf(fp, "Park 95₩n");
    fclose(fp);
    fp = fopen("students.txt", "r");
    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }
    // 데이터를 총 2개씩 정상적으로 읽는 동안만 반복
    while (fscanf(fp, "%19s %d", name, &score) == 2) {
        sum += score;
        count++;
    }
    fclose(fp);

    if (count > 0)
        printf("%.2f₩n", (double)sum / count);
    return 0;
}
```

Binary File I/O

♣ fwrite()

- 메모리에 있는 데이터를 binary 형태로 파일에 쓰는 함수
- fwrite(const void *ptr, size_t size, size_t count, FILE *fp);
- 파일에 쓴 데이터 개수를 반환한다.

```
int n = 100;

//  int 크기의 데이터 1개를 파일에 binary 형태로 저장한다.
fwrite(&n, sizeof(int), 1, fp);
```

ptr : 저장할 데이터의 시작 주소
size : 데이터 하나의 크기
count : 저장할 데이터 개수
fp : FILE 포인터

Binary File I/O

♣ fread()

- binary 파일에서 데이터를 읽어 메모리에 저장하는 함수
- fread(void *ptr, size_t size, size_t count, FILE *fp);
- 파일에서 읽어온 데이터 개수를 반환한다.

```
int n;

// 파일에서 int 크기의 데이터 1개를 읽어 n에 저장한다.
fread(&n, sizeof(int), 1, fp);

printf("%d", n); // n에 들어 있는 값이 출력된다.
```

Binary File I/O

♣ Example: int 배열 저장하기

```
#include <stdio.h>

int main(void) {
    FILE *fp;
    int arr[5] = {10, 20, 30, 40, 50};

    fp = fopen("data.bin", "wb");

    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }

    fwrite(arr, sizeof(int), 5, fp);

    fclose(fp);

    return 0;
}
```

Binary File I/O

♣ Example: int 배열 읽기

```
#include <stdio.h>

int main(void) {
    FILE *fp;
    int arr[5];
    fp = fopen("data.bin", "rb");
    if (fp == NULL) {
        printf("File open failed₩n");
        return 1;
    }
    fread(arr, sizeof(int), 5, fp);

    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }

    fclose(fp);
    return 0;
}
```

Exercise

[Practice] – 1/2

학생 수 N을 입력 받고, N명의 이름과 수학 점수를 입력 받는다. 입력받은 학생 정보를 구조체 배열에 저장한 후, 다음 작업을 수행한다.

1. 학생 정보를 students.bin 파일에 binary 형태로 저장한다.
2. 다시 students.bin 파일을 읽어 학생 정보를 복원한다.
3. 전체 학생 목록, 평균 점수, 최고 점수를 받은 학생을 화면에 출력한다.
4. 같은 결과를 result.txt에 저장한다.

단, 학생 이름은 20자 이하, N은 1 이상이라 가정한다.

Exercise

[Practice] – 2/2

[입력 1]
3
Kim 90
Lee 85
Park 95

[출력 1]
Student List
Kim 90
Lee 85
Park 95

Average: 90.00
Top Student: Park
95

Exercise

[Practice] – Answer(1/2)

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

typedef struct {
    char name[21];
    int score;
} Student;

int main(void) {
    int n;
    Student *students;
    Student *loaded_students;
    FILE *fp;
    FILE *result_fp;
    int sum = 0, top_index = 0;

    scanf("%d", &n);
    students = (Student *)malloc(sizeof(Student) * n);

    if (students == NULL) {
        printf("Memory allocation failed₩n");
        return 1;
    }
    for (int i = 0; i < n; i++) {
        scanf("%20s %d", students[i].name, &students[i].score);
    }
```

```c
    fp = fopen("students.bin", "wb");

    if (fp == NULL) {
        printf("File open failed₩n");
        free(students);
        return 1;
    }
    fwrite(&n, sizeof(int), 1, fp);
    fwrite(students, sizeof(Student), n, fp);

    fclose(fp);
    free(students);
    loaded_students = (Student *)malloc(sizeof(Student) * n);

    if (loaded_students == NULL) {
        printf("Memory allocation failed₩n");
        return 1;
    }
    fp = fopen("students.bin", "rb");

    if (fp == NULL) {
        printf("File open failed₩n");
        free(loaded_students);
        return 1;
    }
```

Exercise

[Practice] – Answer(2/2)

```
fread(&n, sizeof(int), 1, fp);
  fread(loaded_students, sizeof(Student), n, fp);

  fclose(fp);

  for (int i = 0; i < n; i++) {
    sum += loaded_students[i].score;

    if (loaded_students[i].score > loaded_students[top_index].score) {
      top_index = i;
    }
  }
  printf("Student List₩n");
  for (int i = 0; i < n; i++) {
    printf("%s %d₩n", loaded_students[i].name,
loaded_students[i].score);
  }
  printf("₩n");
  printf("Average: %.2f₩n", (double)sum / n);
  printf("Top Student: %s %d₩n",
      loaded_students[top_index].name,
      loaded_students[top_index].score);

  result_fp = fopen("result.txt", "w");
```

```
if (result_fp == NULL) {
    printf("File open failed₩n");
    free(loaded_students);
    return 1;
  }
  fprintf(result_fp, "Student List₩n");
  for (int i = 0; i < n; i++) {
    fprintf(result_fp, "%s %d₩n",
        loaded_students[i].name,
        loaded_students[i].score);
  }

  fprintf(result_fp, "₩n");
  fprintf(result_fp, "Average: %.2f₩n", (double)sum / n);
  fprintf(result_fp, "Top Student: %s %d₩n",
      loaded_students[top_index].name,
      loaded_students[top_index].score);

  fclose(result_fp);

  free(loaded_students);
  loaded_students = NULL;

  return 0;
}
```

THANK YOU!

Any Question?