자료구조

Chap 1-1. C language-Basics (파일 입출력)

2018년 1학기

컴퓨터과학과
민 경 하

Contents

(1) 파일 열기

(2) 파일 입력

(3) 파일 출력

(4) 파일에 대한 임의 접근

(1) 파일 열기

- 파일 포인터

```
void main ( )
{
    FILE *fp;
}
```

(1) 파일 열기

- 파일 열기

```
void main ( )
{
    FILE *fp;
    fp = fopen ( “test.dat”, “r+t” );
}
```

  - fopen (file_name, options)
    - file_name: current directory에 있는 파일만 참조
    - options: “r”, “w”, “a” + “t”, “b”

(1) 파일 열기

- 파일 닫기

```
void main ( )
{
    FILE *fp;
    fp = fopen ( “test.dat”, “r+t” );

    fclose ( fp );
}
```

(1) 파일 열기

- 예제
  - “data.txt”라는 파일을 여는 프로그램을 작성하시오.
  - 만약 이 파일이 없으면 없다는 에러 메시지를 보내도록 하시오.

(1) 파일 열기

- 예제

```
void main ( )
{
    FILE *fp;
    fp = fopen ( “data.txt”, “r+t” );
    if ( fp ) {
        printf(“The file has been successfully opened\n”);
    }
    else {
        printf(“Failed to open the file\n”);
    }
}
```

(1) 파일 열기

- 예제

```
#include <stdio.h>

void main ( )
{
    FILE *fp;
    fp = fopen ( “data.txt”, “r+t” );
    if ( fp ) {
        printf(“The file has been successfully opened\n”);
    }
    else {
        printf(“Failed to open the file\n”);
    }
}
```

(2) 파일 입력

- 파일 입력 함수 (1)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    c = fgetc ( fp );
}
```

  - fgetc (file_name)
    - 파일로부터 하나의 문자를 읽어 들임

(2) 파일 입력

- 파일 입력 함수 (2)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    fgets ( str, n, fp );
}
```

  - fgets (string_name, length, file_name)
    - 파일로부터 length개의 문자를 읽어서 string_name에 저장
    - 또는 개행문자 (\n)이 나타날 때까지 문자를 읽어서 string_name에 저장

(2) 파일 입력

- 파일 입력 함수 (3)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    fscanf ( fp, “%s”, str );
}
```

  - fscanf (file_name, “format”, &args)
    - scanf ( )와 동일한 형식으로 자료를 읽어 들임

(2) 파일 입력

- 예제 1
  - “data.txt”의 내용이 다음과 같을 때, 이 자료를 읽어서 화면에 출력하는 프로그램을 작성한다.

```
5
32.6   21.6
21.0   41.0
56.9   13.4
18.3   8.3
45.1   37.5
```

(2) 파일 입력

```
#include <stdio.h>

void main ( )
{
    FILE *fp;
    if ( (fp = fopen ( “data.txt”, “r+t” )) == NULL ) {
        printf(“Failed to open the file\n”);
        exit ( 0 );
    }

    int n;
    float a, b;
    fscanf ( fp, “%d”, &n );
    for ( int i = 0; i < n; i++ ) {
        fscanf ( fp, “%f%f”, &a, &b );
        printf (“%f\t%f\n”, a, b);
    }

    fclose ( fp );
}
```

(2) 파일 입력

```
#include <stdio.h>

void main ( )
{
    FILE *fp;
    if ( (fp = fopen ( “data.txt”, “r+t” )) == NULL ) {
        printf(“Failed to open the file\n”);
        exit ( 0 );
    }

    int n;
    char str[256];
    fscanf ( fp, “%d”, &n );
    for ( int i = 0; i < n; i++ ) {
        fgets ( str, 256, fp );
        printf (“%s”, str);
    }

    fclose ( fp );
}
```

(2) 파일 입력

- 예제 2
  - “data.txt”의 내용이 다음과 같을 때, 이 자료를 읽어서 화면에 출력하는 프로그램을 작성한다.

```
32.6   21.6
21.0   41.0
56.9   13.4
18.3   8.3
45.1   37.5
```

(2) 파일 입력

- fscanf( )를 사용하는 경우
  - EOF (End Of File)을 읽을 때까지 수행

```
#include <stdio.h>

void main ( )
{
    float a, b;
    FILE *fp;
    if ( (fp = fopen ( “data.txt”, “r+t” )) == NULL ) {
        printf(“Failed to open the file\n”);
        exit ( 0 );
    }

    while ( fscanf (fp, “%f %f”, &a, &b) != EOF )
        printf(“%.1f\t%.1f\n”, a, b);
    fclose ( fp );
}
```

(2) 파일 입력

- fgets( )를 사용하는 경우
  - EOF (End Of File)를 NULL로 인식함

```
#include <stdio.h>

void main ( )
{
    char str[256];
    FILE *fp;
    if ( (fp = fopen ( “data.txt”, “r+t” )) == NULL ) {
        printf(“Failed to open the file\n”);
        exit ( 0 );
    }

    while ( fgets (str, 256, fp) != NULL )
        printf(“%s”, str);
    fclose ( fp );
}
```

(3) 파일 출력

- 파일 출력 함수 (1)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    fputc ( c, fp );
}
```

  - fputc (c, file_name)
    - 파일에 하나의 문자를 씀

(3) 파일 출력

- 파일 출력 함수 (2)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    fputs ( str, fp );
}
```

  - fgets (string_name, file_name)
    - 파일에 string_name에 저장된 문자열을 씀

(3) 파일 출력

- 파일 입력 함수 (3)

```
void main ( )
{
    FILE *fp = fopen ( “data.txt”, “r+t” );

    fprintf ( fp, “%s”, str );
}
```

  - fprintf (file_name, “format”, &args)
    - printf ( )와 동일한 형식으로 자료를 씀

(3) 파일 출력

- 예제 1
  - “data.txt”의 내용이 다음과 같을 때, 이 자료를 읽어서 동일한 내용을 갖는 “data2.txt”라는 파일을 생성한다.

```
32.6   21.6
21.0   41.0
56.9   13.4
18.3   8.3
45.1   37.5
```

(3) 파일 출력

```
#include <stdio.h>

void main ( )
{
    char str[256];
    FILE *ifp, *ofp;
    if ( (ifp = fopen ( “data.txt”, “r+t” )) == NULL )
        printf(“Failed to open the file\n”);
    if ( (ofp = fopen ( “data2.txt”, “w+t” )) == NULL )
        printf(“Failed to open the file\n”);

    while ( fgets (str, 256, ifp) != NULL )
        fprintf(ofp, “%s”, str);

    fclose ( ifp );
    fclose ( ofp );
}
```

(4) 파일에 대한 임의 접근

- 기존의 파일 입출력 함수

```
c = fgetc ( fp );
fgets ( str, n, fp );
fscanf ( fp, “%s”, str );
```

  - 파일에 대한 순차적인 접근
  - 파일의 내용을 읽는 순서를 바꿀 수 없음

(4) 파일에 대한 임의 접근

- 임의 접근 함수
  - rewind ( )
  - fseek ( )
  - ftell ( )

(4) 파일에 대한 임의 접근

- rewind ( )
  - File pointer를 파일의 가장 앞으로 이동
  - fopen ( )을 수행한 직후와 동일한 위치로 이동

```
FILE *fp;
rewind ( fp );
```

(4) 파일에 대한 임의 접근

- rewind ( )
  - 예제 1: rewind ( )를 이용해서 data.txt 파일을 두 번 연속으로 화면에 출력하시오.

```
void main ( )
{
    char str[256];
    FILE *ifp;
    if ( (ifp = fopen ( "data.txt", "r+t" )) == NULL )
        printf("Failed to open the file\n");
    while ( fgets (str, 256, ifp) != NULL )
        printf("%s", str);
}
```

(4) 파일에 대한 임의 접근

- rewind ( )

```
void main ( )
{
    char str[256];
    FILE *ifp;
    if ( (ifp = fopen ( "data.txt", "r+t" )) == NULL )
        printf("Failed to open the file\n");
    while ( fgets (str, 256, ifp) != NULL )
        printf("%s", str);

    rewind ( ifp );  // file pointer를 file의 가장 앞으로 이동

    while ( fgets (str, 256, ifp) != NULL )
        printf("%s", str);
    fclose ( ifp );
}
```

(4) 파일에 대한 임의 접근

- fseek ( )
  - File pointer를 임의의 위치로 이동시킴

```
fseek ( fp, offset, origin );
```

  - origin
    - SEEK_SET: 파일의 맨 처음을 기준으로 적용
    - SEEK_CUR: 현재 위치를 기준으로 적용
    - SEEK_END: 파일의 맨 끝을 기준으로 적용

(4) 파일에 대한 임의 접근

- ftell ( )
  - 다음에 읽거나 쓸 문자의 offset를 반환함

```
    ftell ( fp );
```

(4) 파일에 대한 임의 접근

- fseek ( ) & ftell ( )
  - 예제 2: 파일의 문자 수를 계산하는 프로그램
    - fseek ( )를 이용해서 파일의 맨 뒤로 fp를 이동
    - ftell ( )을 이용해서 파일의 문자 수를 구함

(4) 파일에 대한 임의 접근

```
void main ( )
{
        int ch, n;
        long offset, last;
        FILE *fp;

        fp = fopen ("test.txt", "r+t");
        if ( fp == NULL ) {
                printf("Cannot open file\n");
                exit ( 0 );
        }
//      FILE의 끝으로 이동
        fseek ( fp, 0L, SEEK_END );
//      file 전체에 몇개의 문자가 있는지 셈
        last = ftell ( fp );
        printf("No. of characters including EOF: %d\n", last);
}
```

(4) 파일에 대한 임의 접근

- fseek ( ) & ftell ( )
  - 예제 3: 파일의 문장을 거꾸로 출력하는 프로그램을 구현
    - fseek ( )를 이용해서 파일의 맨 뒤로 fp를 이동
    - ftell ( )을 이용해서 파일의 문자 수를 구함
    - fseek ( )를 이용해서 파일의 맨 뒤에서부터 한문자씩 읽어 들여서 화면에 출력함

(4) 파일에 대한 임의 접근

```
void main ( )
{
        int ch, n;
        long offset, last;
        FILE *fp;

        fp = fopen ("test.txt", "r+t");
        if ( fp == NULL ) {
                printf("Cannot open file\n");
                exit ( 0 );
        }

//      FILE의 끝으로 이동
        fseek ( fp, 0L, SEEK_END );

//      file 전체에 몇개의 문자가 있는지 셈
        last = ftell ( fp );
```

(4) 파일에 대한 임의 접근

```
//      file을 뒤에서 읽어오면서 문자를 읽고 출력함
        for ( offset = 0; offset <= last; offset++ ) {
                fseek ( fp, -offset, SEEK_END );
                ch = fgetc ( fp );
                if ( ch == EOF )
                        continue;
                if ( ch == '\n' )
                        continue;
                printf("%c", ch);
        }
        printf("\n");
        fclose ( fp );
}
```

(4) 파일에 대한 임의 접근

- fseek ( ) & ftell ( )
  - 예제 4: 다음과 같은 파일에서 이름만 출력하는 프로그램을 작성

```
[01] 200712343 LeeSeoJin
[02] 200712344 HanJiMin
[03] 200712345 KimYuJin
```

    - 각 문장에서 이름까지의 offset을 계산

(4) 파일에 대한 임의 접근

```
void main ( )
{
        char name[20];
        FILE *fp;
        fp = fopen ("test2.txt", "r+t");
        if ( fp == NULL ) {
                printf("Cannot open file\n");
                exit ( 0 );
        }
        while ( 1 ) {
                fseek ( fp, 15L, SEEK_CUR );
                if ( fgets ( name, 20, fp ) == NULL )
                        break;
                printf("%s", name);
        }
        fclose ( fp );
}
```

(4) 파일에 대한 임의 접근

- fseek ( ) & ftell ( )
  - 예제 5: 다음의 파일에서 문장을 역순으로 출력하는 프로그램을 작성

```
Red Sox is winning
Yankees are falling
Dodgers are wandering
Giants are drowning
```

- 출력:

```
Giants are drowning
Dodgers are wandering
Yankees are falling
Red Sox is winning
```

(4) 파일에 대한 임의 접근

- fseek ( ) & ftell ( )
  - 예제 5: 다음의 파일에서 문장을 역순으로 출력하는 프로그램을 작성
    - 전체 파일을 읽으면서 각 문장의 길이를 저장함

```
int len[MAX_LINES];
```

    - 파일의 마지막으로 이동
    - 문장의 길이만큼 fseek ( )로 이동하여 문장의 앞으로 이동한 다음 문장을 읽어서 출력함
    - 이 과정을 반복할 것 (2 문장의 길이만큼 앞으로 이동하여야 함)

(4) 파일에 대한 임의 접근

```
void main ( )
{
        int i;
        int cnt;
        long offset, last;
        int len[20];
        char name[256];
        FILE *fp;

        fp = fopen ("test3.txt", "r+t");
        if ( fp == NULL ) {
                printf("Cannot open file\n");
                exit ( 0 );
        }
```

(4) 파일에 대한 임의 접근

```
cnt = 0;
while ( 1 ) {
        if ( fgets ( name, 256, fp ) == NULL )
                break;
        len[cnt++] = strlen(name);
        printf("[%d]%s", len[cnt-1], name);
}
len[cnt] = -1;
```

(4) 파일에 대한 임의 접근

```
        fseek ( fp, 0L, SEEK_END );

        for ( i = cnt; i > 0; i-- ) {
                fseek ( fp, -len[i]-1, SEEK_CUR );
                fseek ( fp, -len[i-1]-1, SEEK_CUR );
                fgets ( name, 256, fp );
                printf("%s", name);
        }
        fclose ( fp );
}
```