การดำเนินการกับแฟ้มข้อมูล (2)

310-230 การโปรแกรมเชิงโครงสร้างและการประยุกต์

File และ Structure

แฟ้มข้อมูลประกอบด้วยข้อมูลซึ่งอาจมีลักษณะ record

ในแต่ละ record จะประกอบด้วยข้อมูลหลาย column หรือ field ที่เหมือนกันหรือสื่อถึงสิ่งเดียวกัน เช่น

รหัสพนักงาน	ชื่อ-สกุล	ตำแหน่ง	หน่วยงาน	เงินเดือน
1	Elvia Aaron	Police Officer	Police	15000
2	Sam Abbott	Electrical Mechanic	Transport	18000
3	Richard Albert	Firefighter	Fire	17500

เราสามารถนำข้อมูลเหล่านี้มาสร้างเป็น structure เพื่อใช้งานในโปรแกรมได้

File และ Structure

รหัสพนักงาน	ชื่อ-สกุล	ตำแหน่ง	หน่วยงาน	เงินเดือน
1	Elvia Aaron	Police Officer	Police	15000
2	Sam Abbott	Electrical Mechanic	Transport	18000
3	Richard Albert	Firefighter	Fire	17500

```
struct personnel {
    int code;
    char name[30];
    char position[40];
    char department[40];
    int salary;
};
```


File และ Structure

ตัวอย่างการอ่านข้อมูลในไฟล์มาเก็บใน structure (ex9_file_struct_read.c)

```
#define LINE_SIZE 80
static const char FILE_NAME[] = "person1.dat";
typedef struct personnel {
        int code;
        char name[30];
        char position[40];
        char department[40];
        int salary;
} PersonData;
PersonData getData(char data[]);
void showPerson(PersonData pdata);
```

กำหนดตัวแปรที่จำเป็นและสร้าง Structure เพื่อรองรับข้อมูลที่อยู่ในไฟล์

File และ Structure

ตัวอย่างการอ่านข้อมูลในไฟล์มาเก็บใน structure (ex9_file_struct_read.c)

```
FILE *fp; int i = 0; char buffer[LINE_SIZE];
PersonData data[10];
fp = fopen(FILE_NAME, "r");
…
do {
   if (fgets(buffer, LINE_SIZE, fp) != NULL) {
      data[i] = getData(buffer);
      showPerson(data[i]);
      i += 1;
   } else { break; }
} while (1); fclose(fp);
```

เปิดไฟล์และอ่านข้อมูลทีละ 1 บรรทัดจะได้ข้อมูลเป็นสายอักขระเช่น

1,Elvia Aaron,Police Officer,Police,15000

File และ Structure

ตัวอย่างการอ่านข้อมูลในไฟล์มาเก็บใน structure (ex9_file_struct_read.c)

```
PersonData getData(char data[]) {
   PersonData pdata; char *p;
   p = strtok(data, ","); pdata.code = atoi(p);
   p = strtok(NULL, ","); strcpy(pdata.name, p);
   p = strtok(NULL, ","); strcpy(pdata.position, p);
   p = strtok(NULL, ","); strcpy(pdata.department, p);
   p = strtok(NULL, ","); pdata.salary = atoi(p);
   return pdata;
}
```

นำข้อมูลได้มาแบ่งออกเป็นส่วน ๆ ด้วย separator ซึ่งในที่นี้คือ (,) ลงใน structure

จาก 1,Elvia Aaron,Police Officer,Police,15000 จะได้เป็น

1 (*integer*)

Elvia Aaron (*char[30]*),

Police Officer (*char[40]*),

Police (*char[40]*),

15000 (*integer*)

File และ Structure

ตัวอย่างการอ่านข้อมูลในไฟล์มาเก็บใน structure (ex9_file_struct_read.c)

```
void showPerson(PersonData pdata) {
   printf("\n#### PERSON ####");
   printf("\nCODE: %d", pdata.code);
   printf("\nNAME: %s", pdata.name);
   printf("\nPOSITION: %s", pdata.position);
   printf("\nDEPARTMENT: %s", pdata.department);
   printf("\nSALARY: %d", pdata.salary);
}
```

แสดงผลข้อมูลบนหน้าจอซึ่งจะเห็นได้ว่าเราสามารถแสดงผลข้อมูลที่อยู่ในไฟล์ในลักษณะของ Structure

File และ Structure

ตัวอย่างการเขียนข้อมูล structure ลงในไฟล์ (ex10_file_struct_write.c)

```
static const char FILE_NAME[] = "person2.dat";
typedef struct personnel {
            int code;
            char name[30];
            char position[40];
            char department[40];
            int salary;
} PersonData;
void writeData(PersonData wdata, FILE *pw);
void showPerson(PersonData pdata);
```

กำหนดตัวแปรที่จำเป็นและสร้าง Structure เพื่อรองรับข้อมูลที่ต้องการเขียนลงในไฟล์

File และ Structure

ตัวอย่างการเขียนข้อมูล structure ลงในไฟล์ (ex10_file_struct_write.c)

```
PersonData newData;
fp = fopen(FILE_NAME, "a+");
newData.code = 4;
strcpy(newData.name, "Jenny Watson");
strcpy(newData.position, "Doctor");
strcpy(newData.department, "Hospital");
newData.salary = 30000;
writeData(newData, fp);
```

ใส่ข้อมูลลงใน structure ที่เตรียมไว้แล้วส่งไปยังส่วนของโปรแกรมสำหรับการเขียนไฟล์

File และ Structure

ตัวอย่างการเขียนข้อมูล structure ลงในไฟล์ (ex10_file_struct_write.c)

```
void writeData(PersonData wdata, FILE *pw) {
fprintf(pw, "%d,%s,%s,%s,%d\n", wdata.code, wdata.name, wdata.position, wdata.department, wdata.salary);
}
```

เขียนข้อมูล structure ลงในไฟล์

การเข้าถึงข้อมูลในไฟล์โดยตรง

ฟังก์ชันเพิ่มเติมสำหรับการเข้าถึงโดยตรง ได้แก่

- rewind
- ftell
- fseek

การเข้าถึงข้อมูลในไฟล์โดยตรง

rewind

ฟังก์ชันสำหรับเปลี่ยน หรือย้ายการทำงานกลับไป ณ จุดเริ่มต้นของแฟ้มข้อมูล

```
void rewind(FILE *stream);
```

การเข้าถึงข้อมูลในไฟล์โดยตรง

rewind (ex11_rewind.c)

```
void showData(FILE *fp);
...
FILE *fp; int close;
fp = fopen("data1.dat", "r");
...
printf("#### DATA in data1.dat ####\n"); showData(fp);
printf("\n##### NO REWIND ####\n"); showData(fp);
rewind(fp);
printf("\n##### REWIND ####\n"); showData(fp);
close = fclose(fp);
```

```
#### DATA in data1.dat ####
LINE: 1000, Jane Foster, 25/03/1990

LINE: 1001, Billy James, 17/06/1991

LINE: 1002, Lars Aldrich, 11/01/1990


#### NO REWIND ####

#### REWIND ####
LINE: 1000, Jane Foster, 25/03/1990

LINE: 1001, Billy James, 17/06/1991

LINE: 1002, Lars Aldrich, 11/01/1990
```


การเข้าถึงข้อมูลในไฟล์โดยตรง

ftell

ฟังก์ชันที่ส่งค่าตำแหน่งปัจจุบันในขณะที่ทำงานกับแฟ้มข้อมูลกลับมาให้ ซึ่งจะเรียกตำแหน่งนี้ว่า relative offset

```
long int ftell(FILE *stream);
```


การเข้าถึงข้อมูลในไฟล์โดยตรง

ftell (ex12_ftell.c)

```
FILE *fp; int close; char buffer[100];
fp = fopen("data1.dat", "r");
...
fgets(buffer, 100, fp);
printf("CURRENT BYTE: %d", ftell(fp));
...
```

```
CURRENT BYTE: 31
```


การเข้าถึงข้อมูลในไฟล์โดยตรง

fseek

ฟังก์ชันใช้ในการเลื่อน หรือเคลื่อนย้ายไปยังตำแหน่งที่ต้องการ เพื่อการอ่าน หรือการเขียนข้อมูลในแฟ้มข้อมูล

```
int fseak(FILE *stream, long int offset, int origin);
```

- offset คือตำแหน่งที่อ้างถึง ซึ่งจะเทียบกับค่า origin
- origin จะมีค่าเป็นไปได้ 3 ค่าคือ
  - มีค่าเป็น 0 หมายถึง จุดเริ่มต้นของแฟ้มข้อมูล (SEEK_SET)
  - มีค่าเป็น 1 หมายถึง ตำแหน่งขณะปัจจุบันของแฟ้มข้อมูล (SEEK_CUR)
  - มีค่าเป็น 2 หมายถึง ตำแหน่งสุดท้ายของแฟ้มข้อมูล (SEEK_END)

การเข้าถึงข้อมูลในไฟล์โดยตรง

fseek

ต้องใช้ origin และ offset ประกอบกันเช่น

- fseek (fp, 0L, 0) หมายถึง ไปยังจุดเริ่มต้นของแฟ้มข้อมูล
- fseek (fp, 10L, 0) หมายถึง ไปยังไบต์ที่ 10 นับจากจุดเริ่มต้นของแฟ้มข้อมูล
- fseek (fp, m, 0) หมายถึง ไปยังไบต์ที่ m นับจากจุดเริ่มต้นของแฟ้มข้อมูล
- fseek (fp, m, 1) หมายถึง ไปยังไบต์ที่ m นับจากตำแหน่งปัจจุบัน
- fseek (fp, -m, 2) หมายถึง ไปยังไบต์ที่ m นับจากท้ายแฟ้มข้อมูล

การเข้าถึงข้อมูลในไฟล์โดยตรง

fseak (ex13_fseek.c)

```
FILE *fp; int close; char buffer[100], c;
...
printf("CURRENT [BYTE: %d], [CHAR: %c]", ftell(fp), fp);
c = fgetc(fp);
printf("\nCURRENT [BYTE: %d], [CHAR: %c]", ftell(fp), c);
fseek(fp, 3L, 1);
c = fgetc(fp);
printf("\nCURRENT [BYTE: %d], [CHAR: %c]", ftell(fp), c);
...
```

```
CURRENT [BYTE: 0], [CHAR:  ]
CURRENT [BYTE: 1], [CHAR: a]
CURRENT [BYTE: 5], [CHAR: e]
```