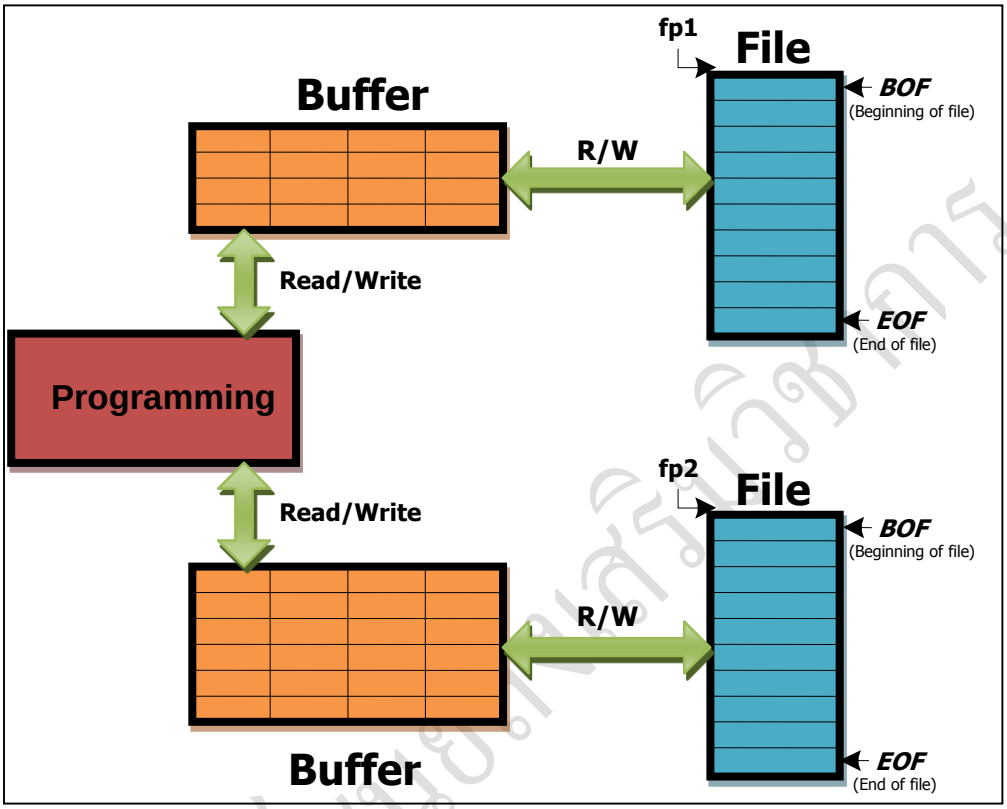
การจัดการแฟ้มข้อมูล

แฟ้มข้อมูลหรือไฟล์คอมพิวเตอร์(File Computer) มีหน้าที่ใช้เก็บข้อมูลที่สัมพันธ์กันหรือเกี่ยวข้องกันรวมเป็นชุดเดียวกัน (ศรัณย์ อินทโกสุม, 2539) เพื่อให้สะดวกในการเรียกใช้และค้นหาข้อมูล ในภาษาซีหรือซีพลัสพลัส(C or C++ Language) กำหนดให้ข้อมูลที่จัดเก็บอยู่ในไฟล์มีลักษณะเป็นข้อมูลที่อยู่ต่อเนื่องกัน ตั้งแต่ต้นไฟล์จนจบไฟล์โดยไม่มีการแบ่งเป็นระเบียน (record) เหมือนกับภาษาระดับสูงทั่วๆไป เช่น ภาษาฟอร์ทแทรน (FORTRAN) ภาษาโคบอล (Cobol) ภาษาปาสคาล (Pascal) ภาษาเบสิก (BASIC) ภาษาวิชวลเบสิก (Visual Basic) และภาษาจาวา (Java) เป็นต้น ในการประมวลผลไฟล์แต่ละครั้งผู้เขียนโปรแกรมจะเป็นผู้กำหนดขอบเขตในการนำข้อมูลไปประมวลผลเอง ในการประมวลผลไฟล์ของภาษาซีโดยทั่วไปทำงาน 2 ลักษณะ คือ

1. การบันทึกข้อมูลลงในไฟล์
2. การอ่านข้อมูลที่เก็บอยู่ในไฟล์ออกมาใช้งาน

การประมวลผลไฟล์ของภาษาซี

วิธีการประมวลผลไฟล์ของภาษาซี สามารถแสดงผังการทำงาน (Diagram Chart) ของขั้นตอนในการประมวลผลไฟล์ชัดเจน (Steven และ Lutfar, 2006) (Brian W. K., Online) ดังนี้



รูปที่ 11.1 ผังการทำงานกับไฟล์

จากรูปที่ 11.1 แสดงผังการทำงานของการไหลข้อมูล เมื่อตัวโปรแกรม (Programming) เขียนให้คอมพิวเตอร์ทำการอ่านหรือเขียน (Read/Write) ระบบจะทำการจองพื้นที่สำรองหรือเรียกว่า บัฟเฟอร์ (Buffer) มาใช้เป็นพื้นที่เพื่อใช้ในการเก็บข้อมูลชั่วคราวที่ทำการติดต่อกับหน่วยความจำของแฟ้มข้อมูล (File) ในที่นี้ตัวชี้ตำแหน่งข้อมูลในไฟล์ชื่อ fp1 และ fp2 ที่ได้ทำการกำหนดตำแหน่งเริ่มต้นของข้อมูล (BOF: Beginning of file) อยู่ที่ตำแหน่ง (Address) ใดของหน่วยความจำ ถึงตำแหน่งท้ายสุดของแฟ้มข้อมูล (EOF: End of file) ซึ่งการไหลของข้อมูลสามารถ ทำการอ่านหรือเขียนลงบนแฟ้มข้อมูลที่กำหนดได้

จากตัวโปรแกรม โดยผ่านบัฟเฟอร์เสมอ ดังนั้นการใช้งานจริง เมื่อไม่ต้องการติดต่อกับแฟ้มข้อมูลแล้ว ผู้โปรแกรม (Programmer) ต้องทำการสั่งให้ยกเลิกการจองหน่วยความจำ (Buffer) ทุกครั้งเสมอก่อนออกจากโปรแกรม เพื่อเป็นการคืนพื้นที่หน่วยความจำให้ระบบ

การบันทึกข้อมูล

ผู้เขียนโปรแกรมสามารถบันทึกข้อมูลลงไปในไฟล์ได้ต่อเมื่อมีการเปิดไฟล์เรียบร้อยแล้ว ซึ่งจะทำให้มีตัวชี้ตำแหน่งข้อมูลในไฟล์ (fp: file pointer) ชี้อยู่ที่ตำแหน่งแรกของไฟล์ โดยตัวชี้ตำแหน่งไฟล์นี้จะใช้บอกถึงตำแหน่งที่กำลังทำการประมวลผลบนไฟล์ให้และในขณะเดียวกันก็จะมีการเตรียมบัฟเฟอร์ ซึ่งเป็นที่เก็บข้อมูลชั่วคราวที่สร้างขึ้นภายในหน่วยความจำ ข้อมูลที่จะทำการประมวลผลจะถูกเก็บไว้ในบัฟเฟอร์ ก่อนบันทึกข้อมูลลงในไฟล์ ข้อมูลจะถูกนำไปเก็บไว้ในบัฟเฟอร์แต่ยังไม่มีการบันทึกลงบนแฟ้มข้อมูล ดังนั้นการบันทึกข้อมูลแต่ละครั้งจะทำให้ตำแหน่งของตัวชี้ตำแหน่งข้อมูลในไฟล์เปลี่ยนแปลงไปตามจำนวนข้อมูลที่บันทึก เมื่อมีการบันทึกข้อมูลจนเต็มบัฟเฟอร์ ระบบจะนำข้อมูลในบัฟเฟอร์นั้น ไปบันทึกในไฟล์ให้เองโดยอัตโนมัติ ในการประมวลผลไฟล์แต่ละครั้ง ระบบสามารถให้เปิดไฟล์ได้มากกว่า 1 ไฟล์ ซึ่งระบบจะทำการจัดเตรียมบัฟเฟอร์ให้เท่ากับจำนวนไฟล์ที่เปิดนั้น โดยมีตัวชี้ตำแหน่งข้อมูลของไฟล์จะอยู่ในตำแหน่งที่แตกต่างกัน ดังรูปที่ 11.1 กรณีที่มีการบันทึกข้อมูลไม่เต็มบัฟเฟอร์ และมีการปิดไฟล์ระบบจะทำการนำข้อมูลที่เหลืออยู่ในบัฟเฟอร์ทั้งหมดบันทึกลงไปในไฟล์ หลังจากนั้นระบบจึงทำการปิดไฟล์

การอ่านข้อมูล

ฟังก์ชันอ่านข้อมูลจากไฟล์ ระบบจะทำการนำข้อมูลมาเก็บไว้ในบัฟเฟอร์ให้ก่อน ก่อนที่จะอ่านข้อมูลจากไฟล์นั้นโดยตรง ในการอ่านข้อมูลในแต่ละครั้งนั้น ระบบจะนำข้อมูลที่ได้จากตำแหน่งที่ตัวชี้ตำแหน่งข้อมูลในไฟล์ที่ชี้อยู่นำไปใช้งาน โดยตำแหน่งของตัวชี้ตำแหน่งข้อมูลในไฟล์ก็จะเปลี่ยนไปตามขนาดของข้อมูลที่อ่านออกมาทำงานจนสิ้นสุดไฟล์ (End of file)

ในการเรียกใช้คำสั่งในการติดต่อกับแฟ้มข้อมูล ภาษาซีได้เตรียมกลุ่มคำสั่งไวในไลบรารี stdio.h ดังนั้นในการเขียนโปรแกรมที่ให้ระบบทำการประมวลผลเกี่ยวกับแฟ้มข้อมูล ผู้เขียนโปรแกรมจำเป็นต้องอ้างไลบรารีด้วยคำสั่ง include <stdio.h> ก่อนเสมอ

การเปิดไฟล์ (Open File)

การเปิดไฟล์แต่ละครั้ง จะต้องเป็นการประกาศการใช้ไฟล์พร้อมกับกำหนดตัวชี้ตำแหน่งข้อมูลไฟล์ สามารถทำการเปิดไฟล์ ทำได้โดยใช้ฟังก์ชัน fopen() ซึ่งอยู่ในไลบรารี stdio.h และตัวชี้ข้อมูลในไฟล์จะถูกกำหนดให้เป็นตัวแปรชนิด FILE โดยประกาศได้ดังนี้

รูปแบบ

```
FILE  *fp;
```

โดย	FILE	เป็นโครงสร้างของข้อมูลใช้ในการสร้างบัฟเฟอร์ (ต้องพิมพ์ ด้วยตัวพิมพ์ใหญ่เสมอ)
	fp	เป็นตัวแปรพอยต์เตอร์ที่ทำหน้าที่ชี้ตำแหน่งของข้อมูลในไฟล์

รูปแบบ

fp = fopen(filename, mode);

โดย filename เป็นชื่อของไฟล์ที่ต้องการเปิด

mode เป็นลักษณะของไฟล์ที่เปิด ตามความหมาย ตารางที่ 11.1

ตารางที่ 11.1 แสดงความหมายชนิดของไฟล์ที่ต้องการเปิดใช้งาน

mode	ความหมาย
“r”	เปิดไฟล์เพื่ออ่านอย่างเดียว
“w”	เปิดไฟล์ใหม่เพื่อเขียนอย่างเดียว หากเป็นไฟล์เดิมจะถูกเขียนทับใหม่
“a”	เปิดไฟล์เพื่อเพิ่มข้อมูล (เขียนต่อท้ายข้อมูลในไฟล์เดิม)
“r+”	เปิดไฟล์ที่มีอยู่แล้วเพื่ออ่านและเขียน
“w+”	เปิดไฟล์ใหม่เพื่ออ่านและเขียน หากเป็นไฟล์เดิมจะถูกเขียนทับใหม่
“a+”	เปิดไฟล์เดิมเพื่ออ่านและเพิ่มข้อมูล ถ้ายังไม่มีจะสร้างไฟล์ใหม่

*หมายเหตุ การเปิดไฟล์แต่ละครั้งอาจจะมีข้อผิดพลาดที่ไม่สามารถเปิดไฟล์ได้จากสาเหตุหลายอย่าง เช่น มีการป้องกันการบันทึกข้อมูลทับ (Write protect) พื้นที่ในการใช้งานเต็ม(Full) หรือมีพื้นที่ไม่เพียงพอ และในกรณีชื่อแฟ้มข้อมูลที่เปิดไม่ถูกต้อง หรือชื่อแฟ้มข้อมูลยาวเกินไป ส่งผลทำให้การส่งกลับจากฟังก์ชัน fopen() จะมีค่าเท่ากับว่าง (Null) หมายความว่า การเรียกใช้ฟังก์ชัน fopen() ทำงานไม่สำเร็จ

การปิดไฟล์ (Close File)

การปิดไฟล์จะทำหลังจากที่สิ้นสุดการใช้งานแฟ้มข้อมูลนั้นแล้ว เพื่อให้นำข้อมูลที่เหลือในบัฟเฟอร์ทั้งหมดเข้าไปบันทึกในไฟล์ ดังนั้นไฟล์ที่มีการเปิดทุกไฟล์จะต้องมีการปิดไฟล์ก่อนสิ้นสุดโปรแกรมเพื่อป้องกันข้อมูลสูญหายของข้อมูล และเป็นการคืนพื้นที่ให้กับหน่วยความจำเมื่อไม่มีการใช้งานแล้ว

รูปแบบ

```
fclose(fp);
```

โดย fp เป็นพอยต์เตอร์ที่ใช้ชี้ตำแหน่งของข้อมูลในไฟล์ที่ต้องการปิด

การปิดไฟล์ถ้าเรียบร้อยดี ด้วยฟังก์ชัน fclose() ระบบจะให้คืนค่าของพอยต์เตอร์เป็นว่างเปล่า(null) แต่ถ้าการปิดแฟ้มข้อมูลไม่เรียบร้อยจะให้ค่าไม่ใช้ว่างเปล่าออกมา โดยปกติระบบการทำงานของภาษาซี จะทำการปิดไฟล์ข้อมูลให้เราโดยอัตโนมัติ แต่เพื่อความถูกต้องและมั่นใจว่าข้อมูลที่เก็บค้างอยู่ที่บัฟเฟอร์ได้ทำการจัดเก็บลงบนแฟ้มข้อมูลเรียบร้อยแล้ว นักเขียนโปรแกรมที่ดีควรจะใช้ฟังก์ชัน fclose() ในการปิดไฟล์เพราะมันเป็นหลักการเขียนโปรแกรมที่ดีนั้นเอง

ชนิดของไฟล์

ด้วยตัวแปลภาษา(Compiler) ของภาษาซีเป็นการทำงานภายใต้ระบบปฏิบัติวินโดว์ของเอ็มเอส-ดอส (MS-DOS) ดังนั้นโค๊ดที่ได้เขียนในตัวโปรแกรม หรือรหัสพิเศษ เช่น \n(new line) ในภาษาซี หมายความว่า เป็นการสั่งให้การแสดงผลหน้าจอภาพเป็นการสั่งขึ้นบรรทัดใหม่ แต่เมื่อนำมาบันทึกลงในไฟล์ที่จะนำมาใช้ใน ของเอ็มเอส-ดอส

จะมีความหมายจะเปลี่ยนไป ดังนั้นเพื่อให้ใช้ไฟล์ที่สร้างขึ้นด้วยภาษาซี สามารถใช้ร่วมกับซอฟต์แวร์ อื่นๆได้ดังนั้นบนของเอ็มเอส-ดอส จึงได้กำหนดชนิดของไฟล์ออกเป็น 2 ชนิด คือไฟล์ชนิดตัวอักษร(Text file) กับไฟล์ชนิดเลขฐานสอง(Binary file) โดยผู้เขียนโปรแกรมจะเป็นผู้กำหนดชนิดของไฟล์ที่ใช้ได้เอง

กล่าวคือถ้าไฟล์ชนิดตัวอักษรมีการบันทึกข้อมูลลงไปในไฟล์ แล้วพบรหัส \n (new line) จะถูกเปลี่ยนเป็น Carriage return และ Line feed ให้แล้วบันทึกลงในไฟล์นั้นให้ ในกรณีที่เป็นการอ่านข้อมูลจากไฟล์ถ้าพบรหัส Carriage return และ Line feed ก็จะเปลี่ยนเป็น new line ให้

เช่นเดียวกับถ้าไฟล์ชนิดเลขฐานสอง ถ้าพบรหัส \n (new line) จะไม่มีการเปลี่ยนเป็น Carriage return และ Line feed ให้ระบบเช่น เพื่อในการทำงานของระบบ จะได้ตีความหมายได้ตรงกับผู้เขียนโปรแกรมต้องการสั่งงานนั่นเอง

ตัวอย่างการเขียนโปรแกรมในการเปิดและปิดไฟล์ที่ถูกต้อง

```
#include<stdio.h>
#define NULL 0                  //กำหนด NULL ให้มีค่าเป็น 0
void main()
{
 FILE *fp;                      //ประกาศตัวแปรพอยน์เตอร์ชื่อ fp สำหรับ
                                ชี้ตำแหน่ง FILE
 fp = fopen(“test.txt”, “w”);   //เปิดไฟล์ใหม่ชื่อ test.txt ชนิดเขียนอย่างเดียว
 …                              //คำสั่งต่างๆ ของการประมวลผลกับไฟล์
 fclose(fp);                    //ปิดไฟล์
}
```

ฟังก์ชัน fopen() จะส่งค่ากลับมาเก็บเป็นค่าที่ชี้ไปที่จุดเริ่มต้นของบัฟเฟอร์มาเก็บไว้ที่ตัวแปรพอยต์เตอร์ชื่อ fp และสุดท้ายหลังจากประมวลผลกับไฟล์เสร็จเรียบร้อยแล้วก็ทำการปิดไฟล์ด้วยฟังก์ชัน fclose() ขอสังเกต การอ้างถึงอาร์กูเมนต์ที่เป็นตัวแปรภายในวงเล็บของฟังก์ชัน fclose(…) จะต้องเป็นชื่อตัวแปรพอยต์เตอร์ที่ชี้จุดเริ่มต้นของบัฟเฟอร์ไฟล์ ที่ประกาศไว้ในตอนแรก หมายความว่า ถ้าตอนเปิดไฟล์ใช้ชื่อตัวแปรพอยต์เตอร์ใด ตอนปิดไฟล์ก็ต้องทำการปิดด้วยชื่อตัวแปรพอยต์เตอร์นั้นเสมอ

ตัวอย่างรูปแบบในการเปิดและปิดไฟล์โดยเพิ่มส่วนตรวจสอบความผิดพลาดในการทำงานของโปรแกรม

```
#include<stdio.h>
#define NULL 0                    //กำหนด NULL ให้มีค่าเป็น 0
void main()
 {
  FILE *fp;                       //ประกาศตัวแปรพอยน์เตอร์ชื่อ fp สำหรับ
                                  ชี้ตำแหน่ง FILE
  fp = fopen("test.txt", "r");    //เปิดไฟล์ชื่อ test.txt เพื่ออ่านอย่างเดียว
  if(fp == NULL)                  //ตรวจสอบว่าไฟล์ที่จะเปิดมีอยู่หรือไม่
    printf("\n Cannot open file"); //ถ้าไม่พบให้แสดง Cannot open file
  else                            //ถ้าพบ NULL ตัวโปรแกรมก็กระทำคำสั่ง
                                  ประมวลผลต่อไป
  …                               //ส่วนของการประมวลผลกับไฟล์
  fclose(fp);                     // ปิดไฟล์
  }
```

จากตัวอย่างโปรแกรมนี้ได้เพิ่มส่วนของการตรวจสอบความผิดพลาดขึ้นมาโดยจะแจ้งให้ทราบ หากการเปิดไฟล์ด้วยฟังก์ชัน fopen() แล้วหาไฟล์ชื่อดังกล่าวไม่พบ หรือฟังก์ชัน fopen() ไม่สามารถเปิดไฟล์ได้สำเร็จ ระบบจะส่งค่ากลับมาเป็นค่า NULL ในโปรแกรมนี้จึงได้ทำการตรวจสอบการเปิดไฟล์ว่าต้องหรือไม่ด้วยคำสั่ง if(fp == NULL) ในการตรวจสอบว่าเปิดไฟล์ไม่สำเร็จก็จะแสดงข้อความ Cannot open file

การประมวลผลโดยใช้ฟังก์ชันมาตรฐานในการรับข้อมูลและแสดงผล (Standard I/O)

ในภาษาซีนั้นการเขียนข้อมูลลงในไฟล์ โดยปกติสามารถเขียนผ่านไฟล์บัฟเฟอร์ (file buffer) ซึ่งก็คือ ส่วนของหน่วยความจำที่กำหนดขึ้นเมื่อเราเปิดไฟล์ และเมื่อข้อมูลในบัฟเฟอร์เต็ม ระบบก็จะมีการเขียนข้อมูลจากไฟล์บัฟเฟอร์ลงไปในไฟล์จริงที่อยู่ในหน่วยความจำ

สำหรับการประมวลผลนั้นจะแบ่งออกได้ 3 ระดับ คือ 1) การประมวลผลระดับตัวอักขระ (Character level) 2) การประมวลผลระดับข้อความ (String level) และ 3) การประมวลผลระดับระเบียน (Record level)

การประมวลผลระดับตัวอักขระ(Character level) เป็นการประมวลผลที่ละหนึ่งตัวอักขระ โดยใช้ฟังก์ชันต่อไปนี้

putc(), fputc()
getc(), fgetc()

ฟังก์ชัน putc(), fputc()

เป็นฟังก์ชันที่ใช้สำหรับบันทึกข้อมูลที่ละหนึ่งตัวอักขระเข้าไปเก็บไว้ในไฟล์ตามตำแหน่งที่ไฟล์พอยต์เตอร์ชี้อยู่

รูปแบบ

```
putc(ch, fp);
```

โดย ch เป็นค่าคงที่ตัวแปรชนิดอักขระที่ต้องการนำข้อมูลเข้าไปเก็บไว้ในไฟล์

fp เป็นตัวชี้ตำแหน่งพอยเตอร์ข้อมูลในไฟล์

ตัวอย่างการเขียนโปรแกรม การรับข้อมูลจากแป้นพิมพ์เข้ามาเก็บในไฟล์ชนิดตัวอักษร (Text File)

```
#include<stdio.h>
void main()
 {
  FILE  *fp;                    //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
  char  ch;                     //ประกาศ ch เป็นตัวแปรชนิดอักขระ
  fp = fopen("data","w");       //เปิดไฟล์ใหม่ชื่อ data ชนิดเขียนอย่างเดียว
      if(fp == NULL)            //ตรวจสอบความผิดพลาด
      {printf("cannot open file\n");     //ถ้าเปิดไม่ได้แสดง cannot open file
      exit(0);                  //ออกจากโปรแกรม
      }
    do{                         //ทำการวนลูปเพื่อรับค่าอักขระมาเขียนลงในไฟล์
```

```
        ch = getchar();          //รับการกดคีย์ 1 อักขระมาเก็บไว้ในตัวแปร
                                 ch
        putc(ch,fp);             //บันทึกข้อมูล 1 อักขระที่เก็บไว้ใน ch ลงใน
                                 ไฟล์
        } while(ch ! = ".");     //หยุดการวนรอบในการเขียนไฟล์เมื่อกด "."
    fclose(fp);                  //ปิดไฟล์
}
```

ฟังก์ชัน getc(), fgetc()

เป็นฟังก์ชันที่ใช้สำหรับอ่านข้อมูลจากไฟล์ที่ละหนึ่งตัวอักขระ จากตำแหน่งที่ไฟล์พอยต์เตอร์ชี้อยู่ ออกมา

รูปแบบ

```
char ch;
ch = getc(fp);
```

โดย fp เป็นตัวชี้ตำแหน่งพอยเตอร์ข้อมูลในไฟล์ของไฟล์ที่จะอ่าน

การอ่านข้อมูลจากไฟล์ถ้าเป็น text file จะอ่านจนจบไฟล์กล่าวคืออ่านจนพบค่า EOF ออกมาเป็นการบอกว่าจบไฟล์แล้ว แต่ถ้าเป็น binary file จะต้องใช้ฟังก์ชัน feof() ตรวจสอบการจบไฟล์

ตัวอย่างโปรแกรมการอ่านข้อมูลจากไฟล์ชนิด text มาแสดงบนจอภาพ

```
#include<stdio.h>
void main()
 {
  FILE  *fp;                          //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
  char  ch;                           //ประกาศ ch เป็นตัวแปรชนิดอักขระ
  fp = fopen("data.dat","r");         //เปิดไฟล์ใหม่ชื่อ data ชนิดเขียนอย่างเดียว
     if(fp == NULL)                   //ตรวจสอบความผิดพลาด
      { printf("cannot open file\n");          //ถ้าเปิดไม่ได้แสดง cannot open file
          exit(0);                    //ออกจากโปรแกรม
      }
    ch = getc(fp);            //อ่านข้อมูลจากไฟล์ตัวแรกมาเก็บไว้ในตัวแปร ch
     while(ch ! = EOF) {  //หยุดอ่านข้อมูลในไฟล์เมื่อพบ EOF
     putchar(ch);             //แสดงข้อมูลตัวอักษรที่อยู่ในตัวแปร ch
     ch = getc(fp);           //อ่านข้อมูลจากไฟล์ 1 อักขระ (ตามรอบการวนลูป)
     }
  fclose(fp);                         //ปิดไฟล์
}
```

การประมวลผลระดับข้อความ (String level) เป็นการประมวลผลที่ละข้อความโดยใช้ฟังก์ชันต่อไปนี้

fputs()

fgets()

ฟังก์ชัน fputs();

เป็นฟังก์ชันที่ใช้บันทึกข้อมูลชนิดข้อความ (string) ลงไปในไฟล์ โดยไม่มีการบันทึกรหัส \0 ซึ่งแสดงการจบข้อความลงในไฟล์

รูปแบบ

fputs (str,*fp);

โดย str เป็นข้อความ(string) ที่ต้องการนำข้อมูลไปบันทึกในไฟล์

fp เป็นตัวชี้ตำแหน่งพอยเตอร์ข้อมูลในไฟล์

ตัวอย่างเช่น fputs (“this is a string”,fp);

หมายความว่า ระบบจะบันทึกข้อความ this is a string นี้เก็บลงไปในไฟล์ที่มี fp เป็นตัวชี้ตำแหน่งพอยเตอร์ในไฟล์ ผลที่ได้จากฟังก์ชันถ้าการบันทึกข้อมูลถูกต้องจะให้ค่า 0 ถ้าการบันทึกไม่ถูกต้องจะให้ค่าที่ไม่ใช่ 0 ออกมา

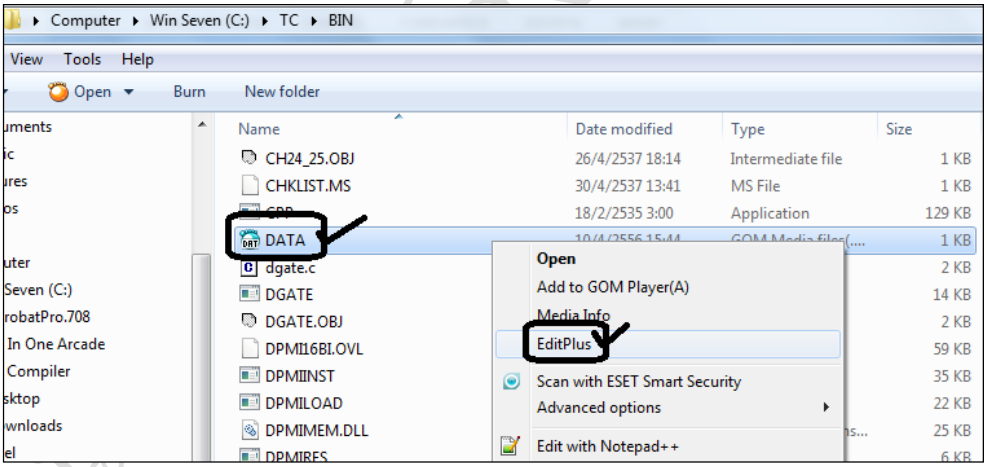
ตัวอย่างโปรแกรมที่ 11.1 การโปรแกรมรับชื่อจากแป้นพิมพ์ 3 ชื่อเข้ามาเก็บไว้ในไฟล์

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
        FILE  *fp;              //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
        char  name[20];         //จองอาร์เรย์ชื่อ name ที่มีสมาชิดจำนวน
                                20 ตัวอักษร
         int i;
         clrscr();
         //ตรวจสอบความผิดพลาดในการอ่านไฟล์
        if((fp = fopen ("data.dat","w"))==NULL) {
            printf("cannot open file\n");
            exit(0);
            }
            for(i=0;i<3;i++)  {          //กำหนดการรับค่า 3 ครั้ง
                printf("Enter name %d : ",i+1);
                gets(name);              //รับค่าข้อความมาเก็บไว้ใน neme
                fputs(name, fp);         //บันทึกข้อมูลใน name ลงในไฟล์
                }
        fclose(fp);
        getch();          }
```

ผลการทำงานโปรแกรมให้ผู้อ่านป้อนข้อความตามตัวอย่างดังนี้

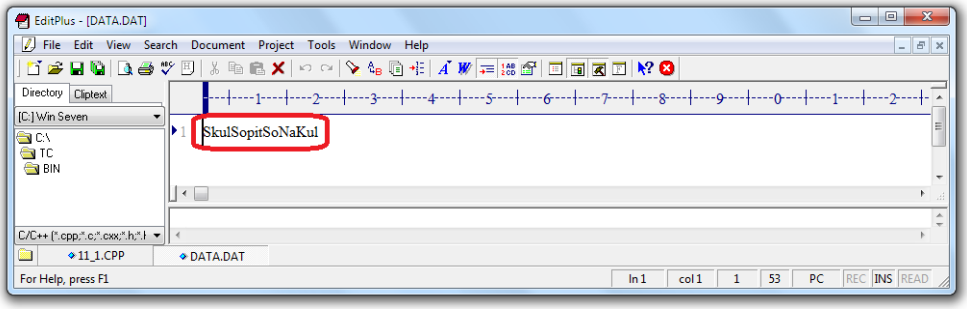
```
Output
Enter name 1 : Skul
Enter name 2 : Sopit
Enter name 3 : SoNaKul
```

ผู้อ่านสามารถตรวจสอบข้อมูลที่เก็บในไฟล์ data.dat โดยให้ผู้อ่านค้นหาไฟล์ที่ได้จัดเก็บไว้ว่าอยู่ที่ใด ในเครื่องคอมพิวเตอร์ของผู้อ่าน จากนั้นให้ทำการเปิดไฟล์ data.dat ด้วยการคลิกเมาส์ปุ่มขวามือ แล้วทำการเลือกเปิดด้วยโปรแกรมอ่านข้อมูล ในที่นี้เลือกเปิดด้วยโปรแกรม EditPlus ตัวอย่างในรูปที่ 11.2



รูปที่ 11.2 แสดงการเปิดไฟล์ data.dat ด้วยโปรแกรม EditPlus

ผลของจัดเก็บข้อความในไฟล์ data.dat จะทำการเก็บอย่างต่อเนื่องกันไป รูปที่ 11.3



รูปที่ 11.3 แสดงการข้อมูลในไฟล์ data.dat มีข้อความว่า SkulSopitSoNaKul

ฟังก์ชัน fgets();

เป็นฟังก์ชันที่ใช้อ่านข้อมูลชนิดข้อความจากไฟล์มาเก็บไว้ในตัวแปรชุด (string) ที่กำหนดโดยสามารถจำนวนอักขระที่อ่านเข้ามาได้

รูปแบบ

fgets(str, num,*fp);

โดย str เป็นตัวแปรชุดที่จะเก็บข้อมูลที่อ่านมาจากไฟล์

num เป็นจำนวนไบท์ของข้อมูลที่จะอ่านเข้ามาแต่ละครั้งเท่ากับ num-1 ตัวหรืออ่านจนพบเครื่องหมาย new line หรือ EOF

fp เป็นตัวชี้ตำแหน่งพอยเตอร์ข้อมูลในไฟล์

ตัวอย่างโปรแกรมที่ 11.2 แสดงฟังก์ชันที่ใช้ fgets() อ่านข้อมูลจาก text file แล้วนำไปแสดงผลบนจอภาพ

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
        FILE  *fp;                  //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
          char  str[128];           //จองอาร์เรย์ชื่อ str ไว้ 128 ตำแหน่ง
          clrscr();
          if((fp = fopen ("data.dat","r"))==NULL) {//ตรวจสอบ Error
            printf("cannot open file\n");
            exit(0);
            }
          while(! feof (fp)) {      //เงื่อนไขการนำข้อมูลออกมาแสดงจนกว่า
                                    จะพบ EOF
            fgets (str,126,fp);     //อ่านข้อมูล string จากไฟล์ออกมาเก็บใน
                                    ตัวแปร str
             printf("%s",str);      //แสดงข้อความที่เก็บในตัวแปร str
            }
                fclose(fp);
}
```

ผลการทำงานโปรแกรม

การประมวลผลระดับระเบียน (Record level)

เป็นการประมวลผลที่ละระเบียน (record) โดยแต่ละระเบียนต้องมีความยาวคงที่ ซึ่งส่วนมากจะใช้กับไฟล์แบบ binary การประมวลผลจะทำการที่ละระเบียนด้วยฟังก์ชันในการประมวลผลต่อไปนี้

fread()

fwrite()

ฟังก์ชัน fread();

เป็นฟังก์ชันที่ใช้ในการอ่านข้อมูลจากไฟล์เข้ามาเก็บไว้ในหน่วยความจำตามตำแหน่งที่กำหนด

รูปแบบ

fread (*ptr, size, num,*fp);

โดย

ptr	เป็นตำแหน่งของหน่วยความจำที่ต้องการนำข้อมูลที่อ่านจากไฟล์เข้ามาเก็บไว้
size	เป็นขนาดของข้อมูลที่อ่านเข้ามาว่ามีขนาดกี่ไบท์
num	เป็นจำนวนระเบียนที่จะอ่านเข้ามาเก็บในหน่วยความจำ
fp	เป็นตัวชี้ตำแหน่งพอยเตอร์ข้อมูลในไฟล์

ค่าที่ได้จากฟังก์ชันจะเป็นจำนวนของค่าที่อ่านเข้ามาตามที่กำหนดไว้ใน num ถ้าการอ่านไม่ได้ตามที่กำหนดหรือพบ end of file ก่อนครบจำนวนที่กำหนดจะเกิดความผิดพลาดขึ้น ดังนั้นจึงควรใช้ฟังก์ชัน feof() ตรวจสอบการจบไฟล์ด้วยในการอ่านข้อมูล

ตัวอย่างโปรแกรม แสดงการอ่านข้อมูล 10 จำนวนมาจากไฟล์ที่ชื่อ balance แล้วนำมาเก็บไว้ในตัวแปรชุดที่ชื่อ bal

```
#include<stdio.h>          #include<conio.h>          #include<stdlib.h>
void main()
{          FILE  *fp;                    //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
float bal[10];                           //จองอาร์เรย์ชื่อ bal ไว้ 10 ตำแหน่ง
if((fp = fopen (“balance”,”r”))==NULL) { //ตรวจสอบความผิดพลาด
                printf(“cannot open file\n”);
                exit(1);
                }
if(fread(bal, sizeof(float), 10, fp) != 10) { //อ่านข้อมูล 10 จำนวนจาก bal
if(feof(fp))          //ตรวจสอบว่าจบไฟล์หรือยัง ถ้าครบ 10 จำนวนแล้ว
printf(“ end-of-file ”);      //แสดงข้อความ end-of-file
          else                           //ถ้าไม่มี
          printf(“ file read error ”);//แสดงข้อความ file read error
               }
            fclose(fp);                  //ปิดไฟล์ที่พอยต์เตอร์ fp ชี้อยู่
          }
```

ฟังก์ชัน fwrite();

ใช้ในการบันทึกข้อมูลจากหน่วยความจำไปเก็บไว้ในไฟล์ในตำแหน่งที่กำหนด

รูปแบบ

fwrite(*ptr, size, num,*fp);

โดย

ptr	เป็นตำแหน่งของข้อมูลภายในหน่วยความจำที่ต้องการนำไปบันทึกลงไฟล์
size	เป็นขนาดของข้อมูลที่จะบันทึกลงในไฟล์ว่ามีขนาดกี่ไบท์
num	เป็นจำนวนระเบียนที่ต้องการบันทึก
fp	เป็นตัวชี้ตำแหน่ง (Pointer) ข้อมูลในไฟล์

ตัวอย่างโปรแกรม แสดงการบันทึกข้อมูลลงในไฟล์ชื่อ test และมีการใช้ฟังก์ชัน sizeof() กำหนดขนาดของข้อมูล

```
#include<stdio.h>
#include<conio.h>
#include<stdlibio.h>
void main()
 {
  FILE  *fp;                          //ประกาศพอยต์เตอร์ชี้ตำแหน่งไฟล์
  float  f = 12.25;                   //กำหนดตัวแปรทศนิยม f มีค่า 12.25
  if((fp = fopen (“test”,”wb”))==NULL) { //ตรวจสอบความผิดพลาด
              printf(“cannot open file\n”);
              exit(1);
              }

fwrite(&f, sizeof (float), 1, fp);    //บันทึกข้อมูลจากตำแหน่ง 12.25 ลงไปใน
                                      //ไฟล์ชื่อ test จำนวน 1 ระเบียน
          fclose(fp);                 //ปิดไฟล์ที่พอยต์เตอร์ fp ชี้อยู่
         }
```

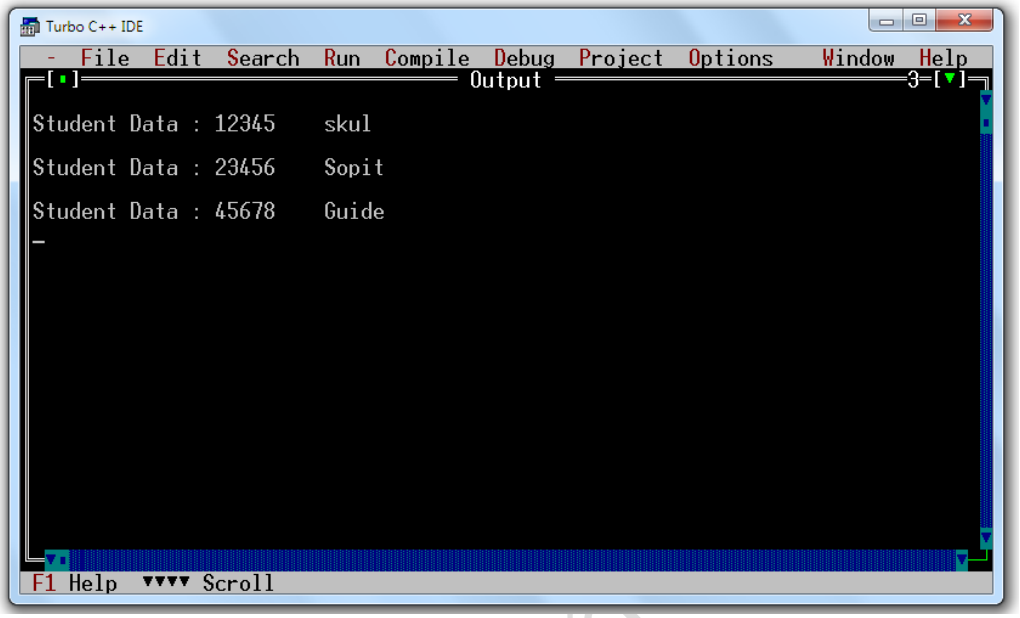
ตัวอย่างโปรแกรมที่ 11.3 แสดงการใช้แฟ้มข้อมูลแบบเรคคอร์ด

```
#include <stdio.h>
#include <conio.h>
#define OK 1
#define ERR 0
#define ARR_SIZE 3
typedef struct {
char ID[8];
char Name[21];
} Student;
int writeData(Student pStudent[ ]) {
                    FILE *fptr;
                    int i;
                    if ((fptr = fopen("student.dat", "w")) != NULL) {
                    for (i = 0; i < ARR_SIZE; i++)
                    fwrite(&(pStudent[i]), sizeof(Student), 1, fptr);
                    fclose(fptr);
                    return (OK);
                    } else
                    return (ERR);                }
int readData(Student *tStudent, int position) {
          FILE *fptr;
```

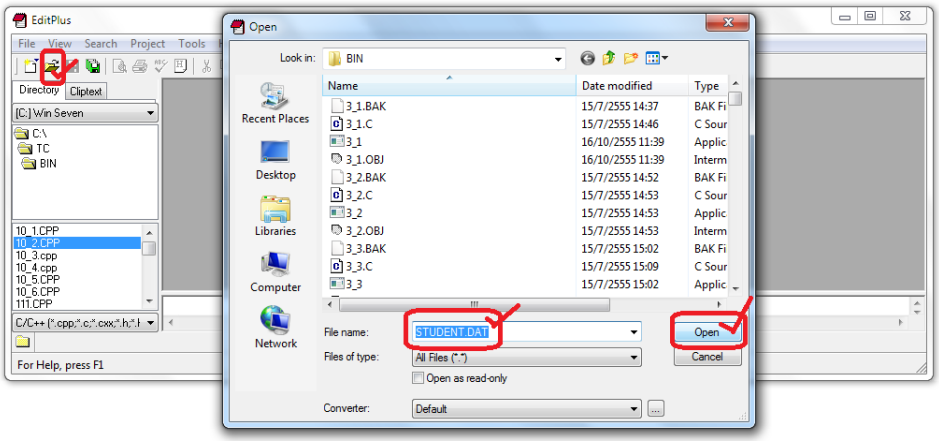
```
 int size;
if ((fptr = fopen("student.dat", "r")) != NULL)
{
        fseek(fptr, 0L, SEEK_END); /* go to end of file */
                size = ftell(fptr);
        if ((position > 0) && (size/sizeof(Student) >= position))
        {
                fseek(fptr, 0L, SEEK_SET); /* go to begin of file */
                fseek(fptr, (position-1)*sizeof(Student), SEEK_SET);
                fread(tStudent, sizeof(Student), 1, fptr);
                printf("\nID %s, Name %s", tStudent->ID,
                tStudent->Name);
                fclose(fptr);
                return (OK);
        } else
        {
        printf("\nYour seek postion is out of range");
        return (ERR);
        }
        } else
        return (ERR);
        }
```

```
void main ( ) { clrscr();
        Student aStudent[ARR_SIZE];
        Student fStudent;
        int i, code, position;
                for (i = 0; i < ARR_SIZE; i++) {
                printf("\nStudent Data : ");
                scanf("%s %s", &(aStudent[i].ID), &(aStudent[i].Name));
                }
        if (code == writeData(aStudent)) {
        printf("\nSelect position : ");
        scanf("%d", &position);
                if (code == readData(&fStudent, position)) {
                printf("\nID : %s, Name : %s", fStudent.ID,
                fStudent.Name);
                                }
                }
}
```

ผลการทำงานโปรแกรม (ให้ผู้อ่านทดลองป้อนข้อมูล รหัสผู้อ่านไม่เกิน 8 หลัก และป้อนชื่อไม่เกิน 21 ตัวอักษร จำนวน 3 คน ดังตัวอย่างด้านล่าง)

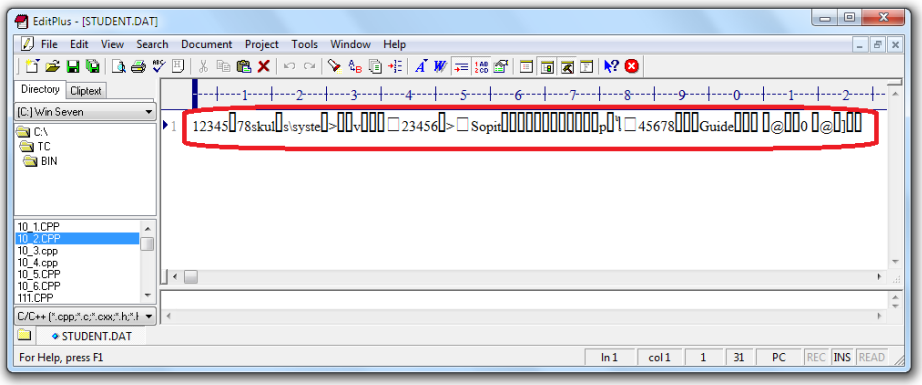


ผู้อ่านสามารถตัวสอบผลการทำงานโปรแกรมที่ได้ทำการเก็บข้อมูลไว้ในไฟล์ student.dat ได้รูปที่ 11.4



รูปที่ 11.4 แสดงที่อยู่ของแฟ้ม student.dat

ผลของการสร้างไฟล์ student.dat ได้ รูปที่ 11.5 ทั้งนี้ข้อความในไฟล์ student.dat จะปรับเปลี่ยนตามที่ผู้อ่านได้ทำการป้อนข้อมูลให้กับโปรแกรม ก่อนหน้านี้



รูปที่ 11.5 แสดงผลการป้อนข้อมูลของผู้ใช้โปรแกรม

จากผลของทำงานโปรแกรมด้านบน พบว่าอาจมีข้อความที่เป็นโค๊ด ของรหัส ASCII บางตำแหน่งที่แสดงออกมาตามการตีความหมายของ ตัวแปลภาษาซี

สรุป

ในการจัดเก็บข้อมูลลงในแฟ้มข้อมูล มีขั้นตอน 3 ขั้นตอน คือ ขั้นตอนแรก ทำการเปิดแฟ้มข้อมูลหรือทำการสร้างแฟ้มข้อมูลใหม่ ในกรณีเป็นการเก็บข้อมูลครั้งแรก ขั้นตอนถัดมาทำการกำหนดข้อมูลลงในแฟ้มข้อมูล โดยระบบจะทำการเก็บข้อมูลไว้ในบัฟเฟอร์หรือที่พักข้อมูลชั่วคราวก่อน ซึ่งเป็นพอยน์เตอร์ที่ถูกกำหนดให้มีพื้นที่ในการจองหน่วยความจำให้เหมาะสม กับข้อมูลให้สามารถรองรับขนาดที่ต้องใช้ในการประมวลผล และขั้นตอนสุดท้าย เป็นขั้นตอนที่ตัวแปลโปรแกรม (Compiler) จะทำการปิดแฟ้มข้อมูล โดย ระบบจะนำข้อมูลที่อยู่ในบัฟเฟอร์มาบันทึกลงในแฟ้มข้อมูลที่กำหนดจากขั้นตอนแรกว่าจะกระทำบนแฟ้มข้อมูลใด

แบบฝึกหัดท้ายบทที่ 11

ตอนที่ 1 จงเติมคำหรือข้อความในช่องว่างต่อไปนี้ให้ถูกต้อง

1. จากรูปแบบของการติดต่อ FILE *fp โดย fp เป็นพอยน์เตอร์ มีค่าเป็นอย่างไร

   .....

2. fp = fopen (filename, mode); อยากทราบว่า mode มีกี่ชนิดประกอบด้วยอะไรบ้าง

   .....

3. จงให้ความหมายของ mode เท่ากับ r คือ

   .....

4. จงให้ความหมายของ mode เท่ากับ w คือ

   .....

5. จงให้ความหมายของ mode เท่ากับ a คือ

   .....

6. จงให้ความหมายของ mode เท่ากับ r+ คือ

   .....

7. จงให้ความหมายของ mode เท่ากับ w+ คือ

   .....

8. จงให้ความหมายของ mode เท่ากับ a+ คือ

   .....

9. การปิดแฟ้มข้อมูลต้องใช้คำสั่งใด

   .....

10. จงอธิบายการปิดแฟ้มข้อมูล

    .....

ตอนที่ 2 จงทำเครื่องหมายกากบาท (x) ทับหน้าข้อที่ถูกต้องที่สุด

1. ฟังก์ชันใดทำหน้าที่ในการบันทึกข้อมูลที่เป็นค่าคงที่ ที่ละตัวลงในแฟ้มข้อมูล

   ก. putc()　　ข. fputc()

   ค. getc()　　ง. fputs()

2. ฟังก์ชันใดทำหน้าที่ในการบันทึกข้อมูลที่เป็นสตริง ลงในแฟ้มข้อมูล

   ก. putc()　　ข. fputc()

   ค. getc()　　ง. fputs()

3. ฟังก์ชันใดทำหน้าที่ในการอ่านข้อมูลที่เป็นสตริง ที่ละตัวจากแฟ้มข้อมูล

   ก. putc()　　ข. fputc()

   ค. getc()　　ง. fputs()

4. ฟังก์ชันใดทำหน้าที่ในการบันทึกข้อมูลที่เป็นสตริง โดยไม่มี \0 ลงแฟ้มข้อมูล

   ก. putc()　　ข. fputc()

   ค. getc()　　ง. fputs()

5. ฟังก์ชันใดทำหน้าที่ในการอ่านข้อมูลจากไฟล์ มาเก็บเป็นตัวแปรสตริง

   ก. fgets()　　ข. fread()

   ค. fwrite()　　ง. fgetc()

6. ฟังก์ชันใดทำหน้าที่ในการอ่านข้อมูลจากไฟล์ มาเก็บเป็นไว้ในหน่วยความจำ

   ก. fgets()　　ข. fread()

   ค. fwrite()　　ง. fgetc()

7. ฟังก์ชันใดทำหน้าที่ในการบันทึกข้อมูลที่หน่วยความจำ ลงแฟ้มข้อมูล

   ก. fgets()　　ข. fread()

   ค. fwrite()　　ง. fgetc()

8. ข้อใดเป็นการตรวจสอบว่าขอมูลหมดแฟ้มหรือยัง

   ก. eof　　ข. exit(1)

   ค. fclose(fp)　　ง. return(ERR)

9. ข้อใดเป็นการสั่งให้ออกจากลูปการตรวจสอบทันที

ก. eof ข. exit(1)

ค. fclose(fp) ง. return(ERR)

10. ข้อใดเป็นการสั่งให้ทำการปิดแฟ้มข้อมูล และทำการคืนหน่วยความจำ

ก. eof ข. exit(1)

ค. fclose(fp) ง. return(ERR)

ตอนที่ 3 จงทำการวิเคราะห์คำถามและทำการเขียนอภิปรายคำตอบตามที่ผู้อ่านเข้าใจโดยยึดความถูกต้องของเนื้อหาประกอบการบรรยาย

1. ให้ผู้อ่านเขียนเพื่อทำการรับค่าสตริงจากแป้นพิมพ์ และทำการบันทึกข้อมูลลงแฟ้มข้อมูลชื่อ store.dat โดยกำหนดให้สตริงที่มีความยาวไม่เกิน 10 ตัวอักษร โดยใช้ฟังก์ชัน fputc()

2. ให้ผู้อ่านเขียนเพื่อทำการคัดลอกแฟ้มข้อมูล store.dat ในข้อ 1. ไปจัดเก็บในแฟ้มข้อมูลใหม่ชื่อว่า stock.dat

3. ให้ผู้อ่านเขียนโปรแกรมเพื่อรับข้อมูลนักศึกษาจำนวน N คน โดยเก็บรายละเอียดข้อมูล รหัสนักศึกษา คณะนักศึกษา และชั้นปีที่ศึกษา โดยบันทึกข้อมูลในแฟ้ม std.dat โดยใช้คำสั่ง fwrite( )

4. ให้นักศึกษาเขียนโปรแกรมเพื่อสร้างตารางแจกแจงความถี่ของจำนวนนักศึกษาโดยแยก ตามคณะและ ชั้นปีที่นักศึกษา โดยอ่านข้อมูลจากแฟ้ม std.dat ในข้อ 3 ดังตัวอย่าง

5. ให้ผู้อ่านเขียนโปรแกรมโดยใช้ประยุกต์ใช้แฟ้มข้อมูลที่ผู้อ่านสนใจมาหนึ่งฐานข้อมูล

เอกสารอ้างอิง

ศรัณย์ อินทโกสุม (2539). ทฤษฎีและตัวอย่างโจทย์การเขียนโปรแกรมด้วยภาษาซี กรุงเทพฯ : แมคกรอฮิล อินเตอร์เนชั่นแนล เอ็นเตอร์ไพรส์, อิงค์.

ธันวา ศรีประโมง (2539). การเขียนโปรแกรมภาษาซีสำหรับวิศวกรรม. พิมพ์ครั้งที่ 4. กรุงเทพฯ : มหาวิทยาลัยเทคโนโลยีมหานคร.

วิจักษณ์ ศรีสัจจะเลิศวาจา และดุษฎี ประเสริฐธิติพงษ์ ออนไลน์ : www.satit.su.ac.th/soottin.

Brian, W. K. Programming in C: A Tutorial Online: http://www.lysator.liu.se/c/bwktutor.html.

Byron S. Gottfried, “Schaum ’s Theory and problems of programming with c” McGraw-Hill,Inc., 1990.

Kenneth A.Barclay. “ANSI C Problem Solving and Programming” Prentice Hall International Ltd., 1990.

Steven, H. & Lutfar, R. (2006). Art of Programming Contest: C Programming | Data Structure | Algorithms (ACM supported), 2nd Edition.