บทที่ 9 การออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อประยุกต์ใช้ในธุรกิจ

สาระการเรียนรู้

- แฟ้มข้อมูลในภาษาซี
- การออกแบบโปรแกรมในลักษณะมอดูลหรือเป็นฟังก์ชันย่อย
- โปรแกรมเมนูหลัก

สมรรถนะการเรียนรู้

1. เขียนโปรแกรมสร้างแฟ้มข้อมูลได้
2. เขียนโปรแกรมเพิ่มข้อมูลแฟ้มใหม่ได้
3. เขียนโปรแกรมอ่านแฟ้มข้อมูลได้
4. เขียนโปรแกรมปรับปรุงแฟ้มข้อมูลได้
5. อธิบายวิธีการออกแบบโปรแกรมในส่วนอินพุตได้
6. อธิบายวิธีการออกแบบโปรแกรมในส่วนประมวลผลได้
7. เขียนโปรแกรมเมนูหลักได้
8. ป้อนและทดสอบโปรแกรมตามตัวอย่างได้
9. เขียนโปรแกรมตามโจทย์ที่กำหนดให้ได้

แผนผังความคิด (Mind Mapping) ของหน่วยการเรียนรู้

แฟ้มข้อมูล

ในการออกแบบโปรแกรมเพื่อประยุกต์ใช้งานธุรกิจนั้น ส่วนที่จำเป็นอย่างหนึ่งก็คือการเขียนโปรแกรมติดต่อกับแฟ้มข้อมูล ดังนั้นในหัวข้อแรกของหน่วยเรียนนี้จึงขอกล่าวในเรื่องของแฟ้มข้อมูล แฟ้มข้อมูลเป็นส่วนที่ถูกจัดเก็บในหน่วยความจำสำรอง การเขียนโปรแกรมให้คอมพิวเตอร์ติดต่อกับหน่วยจัดเก็บข้อมูลสำรองนั้น จะไม่ได้อ่านหรือเขียนข้อมูลไปยังหน่วยความจำสำรองโดยตรง แต่จะใช้พื้นที่ของหน่วยความจำหลักในเครื่องคอมพิวเตอร์จำลองไว้เก็บข้อมูลสำหรับอ่านหรือเขียนแฟ้มข้อมูล โดยมองหน่วยความจำส่วนนี้เสมือนเป็นการอ่านหรือเขียนหน่วยความจำสำรอง เพราะการอ่านหรือเขียนข้อมูลโดยตรงจะใช้เวลามาก ส่งผลให้การทำงานของโปรแกรมช้าไปด้วย ข้อมูลจะถูกบันทึกลงในหน่วยความจำจริงเมื่อมีการสั่งปิดแฟ้มข้อมูลหรือข้อมูลในบัฟเฟอร์นี้เต็มจึงจะมีการบันทึกข้อมูลลงในสื่อหน่วยความจำสำรองจริง

สตรีม (Stream)

เป็นหน่วยของข้อมูลที่เรียงติดกัน โดยอาจเป็นชนิดเดียวกันหรือเป็นโครงสร้างก็ได้สตรีมแบ่งออกเป็น 2 ชนิด คือ

1. เท็กซ์สตรีม (Text Stream)

รูปแบบการเก็บข้อมูลเรียงติดกันในลักษณะของรหัสแอสกี้ (ASCII) และปิดท้ายบรรทัดด้วยรหัส LF : Line Feed และตามด้วยรหัส CR : Carriage-Return ซึ่งเป็นการเลื่อนไปจุดเริ่มต้นของบรรทัดใหม่

2. ไบนารีสตรีม (Binary Stream)

เป็นรูปแบบของการจัดเก็บข้อมูล ซึ่งคงสภาพข้อมูลเดิมไว้ ไม่มีการเปลี่ยนแปลงข้อมูลใด ๆ เมื่อจบบรรทัดจะมีเพียงรหัสขึ้นบรรทัดใหม่ LF : Line Feed เท่านั้น

สตรีมบัฟเฟอร์ (Stream Buffer)

เป็นหน่วยความจำที่ใช้จัดเก็บข้อมูลชั่วคราวสำหรับการอ่านข้อมูลจากแฟ้มข้อมูล หรือส่วนที่เก็บข้อมูลสำหรับนำไปบันทึกลงในแฟ้มข้อมูล

การจัดการแฟ้มข้อมูล

- ในการเขียนโปรแกรมใช้งานแฟ้มข้อมูลในภาษาซี
  จะมีองค์ประกอบที่สำคัญ 5 ส่วน ดังนี้
  - ส่วนนำไฟล์ไลบรารีที่เกี่ยวข้องกับการจัดการไฟล์เข้ามาในโปรแกรม ซึ่งก็คือไฟล์ stdio.h ซึ่งเป็นไฟล์ไลบรารีมาตรฐานเกี่ยวกับอินพุตและเอาต์พุต
  - การประกาศตัวแปรชี้ตำแหน่งไฟล์ หรือไฟล์พอยน์เตอร์(File Pointer) เป็นการจองหน่วยความจำที่ใช้เป็น Stream Buffer โดยใช้คำสั่ง FILE *file_pointer;
  - คำสั่งเปิดไฟล์ ซึ่งเป็นการกำหนดค่าของตัวแปร file pointer ว่ามีชื่อไฟล์และที่อยู่ไฟล์ และมีโหมดการทำงานของไฟล์เป็นแบบใด เพื่อกำหนดให้เป็น Text Stream หรือ Binary Stream และสามารถอ่านหรือเขียนไฟล์ กำหนดเป็นไฟล์ที่สร้างขึ้นมาใหม่ หรือจะเป็นการนำไฟล์เก่ามาปรับปรุงแก้ไขเพิ่มเติม
  - คำสั่งจัดการไฟล์ ซึ่งได้แก่ คำสั่งอ่านไฟล์ หรือเขียนไฟล์
  - คำสั่งปิดไฟล์ เป็นคำสั่งที่ทำการบันทึกข้อมูลจริงลงไปในหน่วยความจำสำรองและปิดแฟ้ม

การเปิดแฟ้มข้อมูล

การใช้งานแฟ้มข้อมูลนั้น สิ่งแรกที่ต้องดำเนินการคือการสร้างสตรีมบัฟเฟอร์ เพื่อใช้เป็นที่เก็บข้อมูลชั่วคราว ด้วยคีย์เวิร์ด FILE *file_pointer

จากนั้นก็จะทำการเปิดแฟ้มข้อมูลด้วยการใช้ฟังก์ชัน fopen()

```
FILE *fpt;
```

รูปแบบ

```
<file_pointer> = fopen(<file_name>,<mode>);
```

เช่น

```
fpt = fopen("textfile.txt", "wt");
```

แสดงโหมดและการทำงานที่ใช้ในฟังก์ชัน fopen ในภาษาซี

โหมด	การทำงาน
r หรือ rt	เปิดแฟ้มข้อมูลแบบเท็กซ์ไฟล์ เพื่ออ่านอย่างเดียว
w หรือ wt	สร้างแฟ้มข้อมูลแบบเท็กซ์ไฟล์ใหม่ เพื่อเขียนอย่างเดียว หากมีไฟล์เก่าอยู่ก็จะเขียนทับด้วยไฟล์ใหม่
a หรือ at	เปิดแฟ้มข้อมูลแบบเท็กซ์ไฟล์ เพื่อเพิ่มข้อมูลต่อท้ายข้อมูลในไฟล์ที่มีอยู่เดิม แต่หากไม่มีไฟล์เก่าก็จะเป็นการสร้างไฟล์ขึ้นมาใหม่
rb	เปิดแฟ้มข้อมูลแบบไบนารีไฟล์ เพื่ออ่านอย่างเดียว
wb	สร้างแฟ้มข้อมูลแบบไบนารีไฟล์ใหม่ เพื่อเขียนอย่างเดียว หากมีไฟล์เก่าอยู่ก็จะเขียนทับด้วยไฟล์ใหม่
ab	เปิดแฟ้มข้อมูลแบบไบนารี เพื่อเพิ่มข้อมูลต่อท้ายข้อมูลในไฟล์ที่มีอยู่เดิม แต่หากไม่มีไฟล์เก่าก็จะเป็นการสร้างไฟล์ขึ้นมาใหม่

แสดงโหมดและการทำงานที่ใช้ในฟังก์ชัน fopen ในภาษาซี

โหมด	การทำงาน
r+ หรือ r+t	เปิดแฟ้มข้อมูลแบบเท็กซ์ไฟล์ที่มีอยู่แล้วเพื่อ อ่านหรือเขียนข้อมูล
w+ หรือ w+t	สร้างแฟ้มข้อมูลแบบเท็กซ์ไฟล์ใหม่ เพื่ออ่านหรือเขียนข้อมูล หากมีไฟล์เก่าอยู่จะถูกแทนที่ด้วยไฟล์ใหม่
a+ หรือ a+t	เปิดแฟ้มข้อมูลแบบเท็กซ์ไฟล์ เพื่อเพิ่มข้อมูลต่อท้ายข้อมูลในไฟล์ที่มีอยู่เดิม แต่หากไม่มีไฟล์เก่าก็จะเป็นการสร้างไฟล์ขึ้นมาใหม่
r+b	เปิดแฟ้มข้อมูลแบบไบนารีไฟล์ที่มีอยู่แล้วเพื่อ อ่านหรือเขียนข้อมูล
w+b	สร้างแฟ้มข้อมูลแบบไบนารีไฟล์ใหม่ เพื่ออ่านหรือเขียน หากมีไฟล์เก่าอยู่ก็จะเขียนทับด้วยไฟล์ใหม่
a+b	เปิดแฟ้มข้อมูลแบบไบนารี เพื่ออ่านหรือเพิ่มข้อมูลต่อท้ายข้อมูลในไฟล์ที่มีอยู่เดิม แต่หากไม่มีไฟล์เก่าก็จะเป็นการสร้างไฟล์ขึ้นมาใหม่

ฟังก์ชัน fopen()

เมื่อเปิดไฟล์ได้จะส่งค่ากลับมาเป็นพอยน์เตอร์ชี้ไปยังจุดเริ่มต้นของบัฟเฟอร์ แต่ถ้าเปิดไฟล์ไม่ได้ จะส่งค่ากลับเป็น NULL ดังนั้นหากจะทำการเปิดไฟล์ที่ไม่ใช่การสร้างแฟ้มใหม่ ควรมีการตรวจสอบก่อนที่จะดำเนินการต่อไป ถ้าเปิดแฟ้มไม่ได้ควรแจ้งข้อผิดพลาดและให้หยุดการทำงาน

เช่น

```
FILE *fpt;
if ((fpt = fopen("test.txt", "rt")) == NULL) {
        printf("Error! Can't Open File\n");
        getch(); exit(1);
}
```

หมายเหตุ การใช้คำสั่ง exit(); จะต้อง #include <stdlib.h> เข้ามาด้วย

การให้บริการแฟ้มข้อมูล

โดยทั่วไปอย่างน้อยจะต้องเขียนโปรแกรมให้สามารถดำเนินการ

- สร้างแฟ้มใหม่ หรือการเขียนแฟ้ม
- อ่านแฟ้มข้อมูล
- เพิ่มข้อมูลในแฟ้ม

ตัวอย่างโปรแกรมสร้างแฟ้มข้อมูลใหม่

```
#include <stdio.h>                    // 1) standard input/output
#include <conio.h>
 void main0
{
                int n,i,ic,pc,sq;
                char  nm[20];         // string
                char  fn[13];         // 8.3
                FILE *fpt;            // 2) file pointer
                clrscr0;printf("New File\n\n");
                printf("Enter Filename : "); scanf("%s",&fn);
                printf("Enter n : "); scanf("%d",&n);
                fpt = fopen(fn,"wt"); // 3) new file
                // r: read, w: write, a: append, t: text, b: binary
                for (i=0; i<n; i++)
                {
                                printf("item code : "); scanf("%d",&ic);
                                printf("name      : "); scanf("%s",&nm);
                                printf("price     : "); scanf("%d",&pc);
                                printf("stock qtt : "); scanf("%d",&sq);
                                printf("%5d %-20s %10d %5d\n",ic,nm,pc,sq);
                                fprintf(fpt,"%5d %-20s %10d %5d\n",
                                                ic,nm,pc,sq); // 4)
                                printf("\n");
                };
                fclose(fpt); // 5)
}
```

ตัวอย่างโปรแกรมเพิ่มข้อมูลต่อแฟ้มเดิม

```
#include <stdio.h>                    // 1) standard input/output
#include <conio.h>
 void main()
{
                int n,i,ic,pc,sq;
                char  nm[20];         // string
                char  fn[13];         // 8.3
                FILE *fpt;            // 2) file pointer
                clrscr();printf("New File\n\n");
                printf("Enter Filename : "); scanf("%s",&fn);
                printf("Enter n : "); scanf("%d",&n);
                fpt = fopen(fn,"at");// 3) append file
                // r: read, w: write, a: append, t: text, b: binary
                for (i=0; i<n; i++)
                {
                                printf("item code : "); scanf("%d",&ic);
                                printf("name      : "); scanf("%s",&nm);
                                printf("price     : "); scanf("%d",&pc);
                                printf("stock qtt : "); scanf("%d",&sq);
                                printf("%5d %-20s %10d %5d\n",ic,nm,pc,sq);
                                fprintf(fpt,"%5d %-20s %10d %5d\n",
                                                ic,nm,pc,sq); // 4)
                                printf("\n");
                };
                fclose(fpt); // 5)
}
```

ตัวอย่างโปรแกรมอ่านข้อมูลจากแฟ้ม

```
// readfile.cpp
// by NuntchayathornChatrsuwun
 #include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct gt // Structure Name Define
{          // 4 Elements in Structure
          int          ic;                    // ID Code
          char         nm[20];    // Name Produce
          int          pc;                    // Price
          int          sq;                    // Stock Quality
}st[500];          // Structure Variable Declaration
```

ตัวอย่างโปรแกรมอ่านข้อมูลจากแฟ้ม (ต่อ)

```
void main()
{
        int n,i;
        char  fn[13];
        FILE *fpt;
        clrscr();printf("File Reader Program\n\n");
        printf("Enter filename : "); scanf("%s",&fn);
        if ((fpt = fopen(fn,"rt")) == NULL)
        {
                printf("Cannot Open file! \n\n");
                getch(); exit(1);
```

ส่วนแรกของฟังก์ชันหลัก จะแสดงหัวโปรแกรมและเปิดแฟ้มข้อมูลโดยรับชื่อแฟ้มทางแป้นพิมพ์ ถ้าไม่มีแฟ้มนี้อยู่ก็จะจบโปรแกรม

ตัวอย่างโปรแกรมอ่านข้อมูลจากแฟ้ม (ต่อ)

```
i=0;
while (! feof(fpt)) // eof : End of File
{
        fscanf(fpt,"%d %s %d %d\n",
          &st[i].ic,&st[i].nm,&st[i].pc,&st[i].sq);
        printf("%3d %-20s %10d %5d\n",
          st[i].ic,st[i].nm,st[i].pc,st[i].sq);
        i++; // count item
};
fclose(fpt);  getch();
```

หลังจากที่เปิดแฟ้มได้แล้ว ก็จะเข้าสู่วงวนหรือลูป while เพื่อทำการอ่านข้อมูลจากแฟ้มมาทีละบรรทัด ด้วยฟังก์ชัน fscanf(); มาเก็บไว้ในตัวแปรโครงสร้าง และทำการนับจำนวนเรคอร์ดที่อ่านเข้ามา เมื่ออ่านข้อมูลทั้งหมด โดยฟังก์ชัน feof(); จะตรวจสอบว่าสิ้นสุดไฟล์หรือไม่ถ้าอ่านจนจบไฟล์ก็จะออกจากลูป while แล้วปิดแฟ้มข้อมูล และรอการกดคีย์ใด ๆ เพื่อทำงานต่อไป

ตัวอย่างโปรแกรมอ่านข้อมูลจากแฟ้ม (ต่อ)

```
        n=i;
        printf("\nNumber of Items = %d\n",n);
        for (i=0; i<n; i++)
        {
                printf("%3d %-20s %10d %5d\n",
                   st[i].ic,st[i].nm,st[i].pc,st[i].sq);
        };
        getch0;
}
```

ส่วนท้ายของโปรแกรมหลัก จะเป็นการรายงานผลข้อมูลที่ถูกเก็บไว้ในตัวแปรโครงสร้าง ทั้งหมดออกมา

การออกแบบโปรแกรม

อาจแบ่งออกเป็น 3 ส่วน

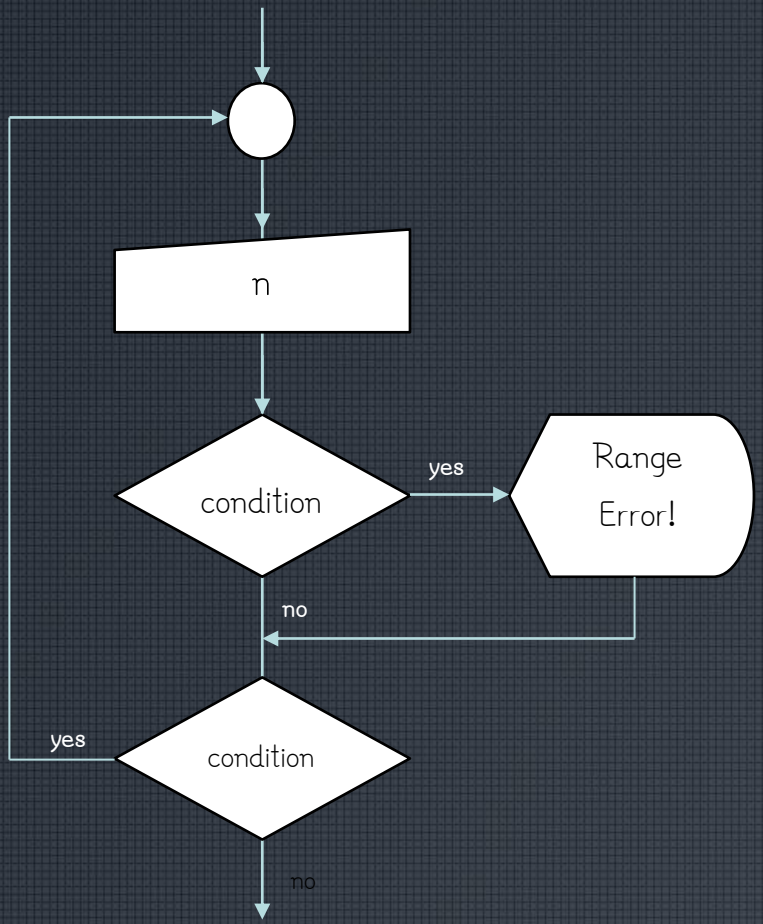
1. ส่วนอินพุต
2. ส่วนประมวลผล
3. ส่วนแสดงผล

การออกแบบส่วนอินพุต

การรับข้อมูลจากแป้นพิมพ์ จะเหมาะกับการรับค่าจำนวนน้อย ๆ ส่วนการอ่านข้อมูลจากแฟ้มข้อมูล จะเหมาะกับการรับค่าจำนวนมาก ซึ่งได้จัดเตรียมไว้ล่วงหน้า ผู้ใช้ไม่ต้องเสียเวลาที่จะต้องป้อนค่าใหม่ทุกครั้ง และยังสามารถปรับปรุงข้อมูลในแฟ้มข้อมูล และนำข้อมูลกลับมาใช้ใหม่ได้ การอ่านข้อมูลจะใช้ตัวแปรชุดหรือตัวแปรโครงสร้างสำหรับจัดเก็บข้อมูล โครงสร้างโปรแกรมในการรับค่าข้อมูลจากแป้นพิมพ์หรืออ่านจากแฟ้มข้อมูล

การเขียนโปรแกรมในส่วนอินพุต จะมุ่งเน้นตรวจสอบข้อมูลที่ถูกต้อง เช่น ข้อมูลมีช่วงระหว่าง 0-100 หากมีการป้อนข้อมูลไม่ได้อยู่ในช่วงที่กำหนด ควรมีการรายงานผลข้อผิดพลาด (หรืออาจมีเสียงเตือน) และให้ป้อนข้อมูลใหม่การเขียนโปรแกรมในลักษณะนี้เป็นการแก้ปัญหา Runtime Error ได้ในระดับหนึ่งเท่านั้น

การออกแบบส่วนอินพุต




```
do
{

   printf("Enter N : ");
   scanf("%d",&n);

if (condition)
   {
            printf("Range Error! \n");
            printf("Please Enter 0-100\n");
            fflush(stdin);
   }


} while (condition);
```

การเขียนเงื่อนไข ในส่วนตรวจสอบข้อมูลอินพุต

การเขียนเงื่อนไข (Condition) เพื่อตรวจสอบเงื่อนไขตามตัวอย่างนี้ อยู่ระหว่าง 0-100 สามารถเขียนเงื่อนไขได้ 2 แบบ คือ

- (n < 0 || n > 100)
- (!(n >= 0 && n <= 100))

การป้อนค่าข้อมูลในคำสั่ง scanf() นั้น ถ้าหากป้อนค่าผิดชนิด ในตัวอย่างจะต้องป้อนค่าที่เป็นตัวเลข แต่ถ้าผู้ใช้ป้อนค่าเป็นข้อความหรือตัวอักขระ ก็จะเกิดข้อผิดพลาดขึ้น การแก้ปัญหานี้ จึงต้องเขียนโปรแกรมตรวจสอบข้อผิดพลาด และแจ้งข้อผิดพลาดที่เกิดขึ้น และสิ่งที่ขาดไม่ได้ของการแก้ปัญหานี้คือการเคลียร์บัฟเฟอร์ของหน่วยความจำในการรับข้อมูลทางแป้นพิมพ์ คือใช้ฟังก์ชัน fflush(stdin); ก็จะสามารถแก้ปัญหา Runtime Error ที่จะเกิดขึ้นระหว่างการรันโปรแกรมภาษาซี

โปรแกรมรับค่าอินพุตที่มีการตรวจสอบค่า ในช่วง 0-100

```
// p192-1.cpp
#include <stdio.h>
#include <conio.h>
void main0
{
        int n;
        clrscr0;
        printf("Input Program\n\n");
        do {
                printf("Enter n : "); scanf("%d",&n);
                if (n < 0 || n > 100) {
                        printf("\nRange Error !!!\n");
                        printf("Please enter 0-100\n\n");
                        fflush(stdin);
                }
        } while (n < 0 || n > 100);
        printf("Input number is %d\n",n);
        printf("Pass OK\n");
        getch0;
}
```

สร้างฟังก์ชันรับค่าอินพุตที่มีการตรวจสอบค่าในช่วง 0-100

```
#include<stdio.h>
#include<conio.h>
int input0;  // Function Declaration
void main0
{
                int n;
                clrscr0;
                printf("Input Program\n\n");
                n = input0;          // Call User Define Function
                printf("Input number is %d\n",n);
                printf("Pass OK\n");
                getch0;
}
int input0  // User Define Function Detail
{
                int n;
                do {
                                printf("Enter n : ");  scanf("%d",&n);
                                if (n < 0 || n > 100) {
                                                printf("\nRange Error !!!\n");
                                                printf("Please enter 0-100\n\n");
                                                fflush(stdin);
                                }
                } while (n < 0 || n > 100);
                return n;
}
```

การออกแบบส่วนประมวลผล

ส่วนของการประมวลผลนั้น มี 4 ลักษณะ คือ

1. การประมวลผลโดยใช้สูตร (Formula)
2. การประมวลผลโดยการทำซ้ำ (Repeation)
3. การเปิดตาราง (Lookup table)
4. ฟังก์ชันเรียกตัวเอง (Recursive Function)

การประมวลผลโดยใช้สูตร (Formula)

ซึ่งเป็นวิธีแบบพื้นฐานของการเขียนโปรแกรมโครงสร้างแบบลำดับ ดังได้กล่าวมาแล้วในบทต้น ๆ วิธีการนี้มีข้อจำกัดที่สูตร จึงไม่ยืดหยุ่น เป็นทางเลือกแรกที่นำมาเขียนโปรแกรม ถ้าไม่มีสูตรหรือคิดสูตรไม่ได้ จึงค่อยพิจารณาข้อต่อไป

การประมวลผลโดยการทำซ้ำ (Repeation)

วิธีการนี้สามารถเขียนโปรแกรมได้ยืดหยุ่น มีข้อเสียที่จำนวนการทำซ้ำเมื่อมีไม่เท่ากัน จะทำให้การคำนวณใช้เวลาต่างกันไปด้วย คือถ้าจำนวนลูปมีมากก็จะใช้เวลามาก ถ้าจำนวนลูปมีน้อยก็จะใช้เวลาน้อยกว่าซึ่งการประมวลผลโดยการทำซ้ำนี้เป็นวิธีที่ส่วนใหญ่ใช้ในการเขียนโปรแกรมคอมพิวเตอร์ เพราะโปรแกรมจะสั้นและมีความยืดหยุ่นสูง สิ่งที่ต้องคำนึงถึงการใช้วิธีการประมวลผลแบบนี้ก็คือเทคนิคหรือวิธีการที่จะทำให้การวนรอบหรือลูปน้อยที่สุดหรือลดลง เพื่อที่จะให้การคำนวณหาคำตอบได้เร็วที่สุด

โปรแกรมหาค่า Factorial n! โดยใช้วิธีการแบบทำซ้ำ

```
// p194.cpp
#include<stdio.h>
#include<conio.h>
double fact(unsigned int);
void main0
{
        unsigned int i;
        clrscr0;
        printf("Factorail n!\n\n");
        for (i=1; i<=20; i++)
                printf("%3u! = %35.0lf\n",i,fact(i));
        getch0;
}
double fact(unsigned int n)
{
        double f=1;
        unsigned int i;
        for (i=1; i<=n; i++)
                f*=i;
        return f;
}
```

การเปิดตาราง (Lookup table)

วิธีการนี้จะทำการจัดเก็บคำตอบไว้ในรูปตาราง คือจัดเก็บไว้ในตัวแปรแบบอาร์เรย์ เมื่อรับค่าอินพุตเข้ามาก็จะนำค่านั้นเป็นเปิดตาราง แล้วนำคำตอบจากตารางมาใช้งาน จึงทำให้ใช้เวลาที่จะได้คำตอบเท่ากัน ข้อเสียคือมีข้อจำกัดที่การจัดเก็บคำตอบที่อาจมีจำกัดและเสียเวลาในการจัดเก็บคำตอบมาก หากจำนวนความถี่ที่ใช้งานไม่มากพอ อาจใช้เป็นทางเลือกสุดท้ายในการตัดสินใจในการเลือกวิธีนี้ ตัวอย่างของวิธีนี้ ได้แก่ การจัดเก็บตารางทางสถิติต่าง ๆ ในโปรแกรม โดยข้อมูลตารางอาจจะอยู่ภายในโปรแกรม หรืออาจจัดเก็บไว้ในรูปของแฟ้มข้อมูลก็ได้ ซึ่งวิธีหลังจะค่อนข้างสะดวกกว่า เป็นวิธีที่นิยมใช้กับโปรแกรมตารางคำนวณเป็นส่วนใหญ่

โปรแกรมหาค่า Factorial n! โดยใช้วิธีการเปิดตาราง

```
#include<stdio.h>
#include<conio.h>
double fact(unsigned);
void main()
{
        unsigned int i;
        clrscr(); printf("Factorail n!\n\n");
        for (i=1; i<=20; i++)
                printf("%3u! = %35.0lf\n",i,fact(i));
        getch();
}
double fact(unsigned n)
{
        double f[]={
        1.0,1.0,2.0,6.0,24.0,120.0,720.0,5040.0,40320.0,362880.0, 3628800.0,39916800.0,479001600.0,
        6227020800.0,87178291200.0, 1307674368000.0,20922789888000.0,355687428096000.0, 6402373705728000.0,
        121645100408832000.0,  2432902008176640000.0,51090942171709400000.0, 1124000727777610000000.0,
        25852016738885000000000.0, 620448401733239000000000.0,15511210043331000000000000.0,
        403291461126606000000000000.0, 10888869450418400000000000000.0, 304888344611714000000000000000.0,
        8841761993739700000000000000000.0, 265252859812191000000000000000000.0};
        return f[n];
}
```

ฟังก์ชันเรียกตัวเอง (Recursive Function)

วิธีการที่ใช้ฟังก์ชันเรียกตัวเองนั้น จะค่อนข้างอธิบายการทำงานได้ยาก แต่วิธีนี้จะเขียนโปรแกรมได้สั้น ตัวอย่างที่ต้องใช้วิธีการเขียนแบบฟังก์ชันเรียกใช้ตัวเอง (Recursive Function) เช่น โปรแกรมการจัดเรียงข้อมูลแบบ Quick Sort ซึ่งถือว่าเป็นวิธีการจัดเรียงข้อมูลที่ประมวลผลได้เร็วที่สุด

โปรแกรมหาค่า Factorial n! โดยใช้วิธีการเรียกใช้ตัวเอง

```
// recur.cpp
#include<stdio.h>
#include<conio.h>
double fact(unsigned);
void main()
{
        unsigned int i;
        clrscr();       printf("Factorail n!\n\n");
        for (i=1; i<=20; i++)
        {
                printf("%3u! = %35.0lf\n",i,fact(i));
        }
        getch();
}
double fact(unsigned n)
{
        if (n == 0)
    return 1;
        else
    return (n * fact(n-1));
}
```

การออกแบบส่วนเอาต์พุต

การออกแบบส่วนเอาต์พุตมุ่งเน้นการนำผลลัพธ์ที่ได้จากการประมวลผลแสดงผลออกทางจอภาพส่วนหนึ่ง หรือจัดเก็บในรูปแบบของแฟ้มข้อมูล การรายงานผลทางจอภาพจะมีลักษณะที่เป็นตัวอักขระในรูปแบบตาราง หรือแสดงในรูปแบบของกราฟต่าง ๆ เพื่อให้ดูเป็นระเบียบ สวยงาม ง่ายต่อความเข้าใจ โดยอาจมีส่วนหัวตารางหรือหัวเรื่อง และตามด้วยรายละเอียดของข้อมูลในแต่ละคอลัมน์ สำหรับการแสดงผลทางจอภาพจะมีข้อจำกัดที่จำนวนบรรทัด เช่น ในเท็กซ์โหมดของโปรแกรมภาษาซี จะสามารถแสดงผลได้เพียง 80 อักขระต่อหนึ่งบรรทัด และมีจำนวน 25 บรรทัด ดังนั้นในรายละเอียดของข้อมูลในตารางเมื่อหักลบหัวเรื่องและหัวตารางและส่วนท้ายตาราง จะสามารถรายงานผลได้เพียงประมาณ 20 บรรทัด ถ้ามีข้อมูลจำนวนบรรทัดมากกว่า การเขียนโปรแกรมจะจำกัดให้แสดงผลเพียง 20 บรรทัด โดยการแสดงทีละหน้าจอ (Page) แล้วให้หยุดรอการกดคีย์ เมื่อกดคีย์เพื่อเลื่อนไปหน้าจอถัดไป ซึ่งอาจมีการกดคีย์อื่น ๆ เพื่อควบคุมให้เลื่อนหน้าจอขึ้นหรือลง หรือเลื่อนไปทางขวาเพื่อดูข้อมูลที่มีมากกว่า 80 คอลัมน์ และเลื่อนไปทางซ้าย เพื่อย้อนกลับ

การออกแบบโปรแกรมเมนูหลัก (Main Menu)

การออกแบบโปรแกรมที่มีการรวมงานหรือรวมหลายโปรแกรมมาไว้เป็นโปรแกรมเดียวกัน สามารถทำได้โดยการออกแบบลักษณะเป็นโปรแกรมย่อยรวมไว้ในโปรแกรมเดียวกัน (ในภาษาซีจะเขียนในลักษณะเป็นฟังก์ชันย่อย) หรืออาจแยกโปรแกรมแล้วใช้วิธีเรียกใช้งานจากโปรแกรมเมนูหลัก

การเรียกไฟล์โปรแกรมที่ผ่านการคอมไพล์เป็นไฟล์ .EXE นั้น ในภาษาซีจะใช้คำสั่ง ดังนี้

```
system("file.exe");
```

file.exe หมายถึงไฟล์โปรแกรมที่ผ่านการคอมไพล์เป็น .EXE ที่สามารถรันได้แล้ว

ซึ่งฟังก์ชัน system() ถูกนิยามไว้ในไฟล์ stdlib.h จึงต้องทำการ #include ไฟล์นี้เข้ามาในส่วนหัวโปรแกรม แต่มีขั้นตอนการรันโปรแกรมไม่เหมือนกับโปรแกรมปกติ คือ หลังจากคอมไพล์โปรแกรมแล้ว ต้องออกไปเรียกใช้งานใน DOS Prompt เท่านั้น จึงจะรันโปรแกรมได้

โปรแกรมเมนูหลัก

```
// Program : menu.cpp
// By      : Nuntchayathorn No.99
// Date    :
#include <conio.h>
#include <stdio.h>

void p10;
void p20;
void p30;
```

ส่วนหัวของโปรแกรมจะเป็นการประกาศฟังก์ชันที่ใช้เป็นโปรแกรมย่อยต่าง ๆ

โปรแกรมเมนูหลัก (ต่อ)

```
void main0
{
        char ch;
        do {
                do {
                        clrscr0;
                        printf("----------------------\n");
                        printf("      Main Menu \n");
                        printf("----------------------\n");
                        printf("  1) Program 1 \n");
                        printf("  2) Program 2 \n");
                        printf("  3) Program 3 \n");
                        printf("  0) Exit      \n");
                        printf("----------------------\n");
                        printf("Please Select 0-3 : ");
                        ch = getche0;
                } while ((ch<'0' || ch>'3') && ch!=27);
```

ในส่วนแรกของฟังก์ชันหลักจะเป็นการแสดงรายการเมนู แล้วรอให้มีการกดเลข 0-3 เพื่อเลือกรายการในเมนูที่กำหนด หรือกดคีย์ Esc เพื่อออกจากโปรแกรม

โปรแกรมเมนูหลัก (ต่อ)

```
            printf("\n\n");
            switch (ch)
            {
                        case '1' : p10; break;
                        case '2' : p20; break;
                        case '3' : p30; break;
                        case '0' : ch = 27;
            }
      } while (ch != 27);
      printf("Bye... bye...\n"); getch0;
 }
```

หลังจากที่ได้เลือกรายการแล้ว ส่วนนี้จะทำการตรวจสอบเพื่อเลือกฟังก์ชันที่จะทำงานต่อไป โดยตรวจสอบด้วยคำสั่ง switch/case โดยถ้าเป็นการกดคีย์ Esc หรือ เลือกรายการ 0 จากเมนู ก็จะเป็นการออกจากโปรแกรม แต่ถ้าเลือกรายการที่ 1-3 ก็จะเป็นการเรียกโปรแกรม p10; p20; p30; ตามลำดับ

ฝึกปฏิบัติ

- ให้นักเรียนได้ศึกษาโปรแกรมๆ ฝึกป้อนโปรแกรมจากหนังสือเรียนทุกตัวอย่าง

จบการนำเสนอ

บทที่ 9
การออกแบบและเขียนโปรแกรมอย่างง่ายเพื่อประยุกต์ใช้ในธุรกิจ