บทที่ 5(ต่อ) ฟังก์ชัน (Function)

5.5 ฟังก์ชันแบบเรียกซ้ำ

ฟังก์ชันแบบเรียกตัวเอง เป็นฟังก์ชันที่สามารถกำหนดรูปแบบของฟังก์ชันในเทอมใด ๆ กับค่าของเทอมในก่อนหน้าได้ในรูปแบบที่แน่นอน โดยที่จะต้องมีจุดสิ้นสุดของฟังก์ชันที่แน่นอน ตัวอย่างของฟังก์ชันที่สามารถทำงานในรูปของฟังก์ชันแบบเรียกตัวเองได้ เช่น ฟังก์ชันแฟคทอเรียล (Factorial)

```
0 ! = 1
1 ! = 1
2 ! = 2 * 1
3! = 3 * 2 * 1
```

ซึ่งสามารถเขียนฟังก์ชันแฟคทอเรียลในรูปแบบของความสัมพันธ์ได้ดังนี้

n ! = 1 ในกรณี n = 0

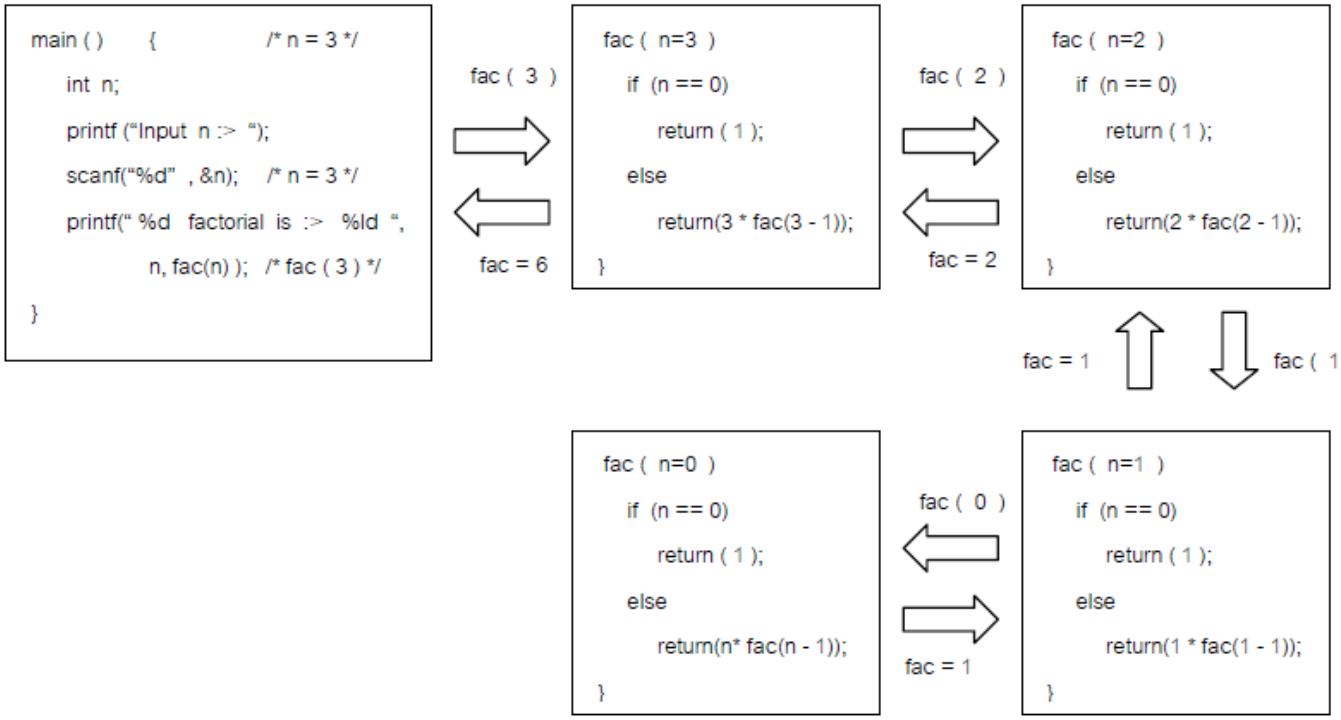
n ! = n * (n-1) * (n-2) * ... * 3 * 2 * 1 ในกรณีที่ n >0

กรณีที่ n = 0 เป็นจุดสิ้นสุดของการเรียกตัวเอง ในกรณีของแฟคทอเรียลฟังก์ชัน แสดงโปรกแกรมดังตัวอย่างที่ 5.6

<u>ตัวอย่างที่ 5.6</u> แสดงแฟคทอเรียลฟังก์ชัน ในลักษณะของฟังก์ชันเรียกตัวเอง

```
#include <stdio.h>
long int fac ( int n ) {
    if (n == 0)
        return ( 1 );
    else
        return ( n * fac(n-1) );
}
main ( ) {
    int n;
    printf (“Input n :> “);
    scanf(“%d” , &n);
    printf(“ %d factorial is :> %ld “, n, fac(n) );
}
```

สมมติให้ n = 3 สามารถแสดงการทำงานได้ดังรูปที่ 5.9



รูปที่ 5.9 แสดงการทำงานของแฟคทอเรียลฟังก์ชัน

การเขียนฟังก์ชันเรียกตัวเองจะเหมาะกับงานที่มีลักษณะเฉพาะเท่านั้น การเขียนฟังก์ชันในลักษณะเรียกตัวเองช่วยให้อ่านโปรแกรมได้เข้าใจมากขึ้น แต่จะทำให้เปลืองทรัพยากรในการทำงานเพิ่มขึ้นและการทำงานช้ากว่าโปรแกรมที่เขียนในลักษณะปกติ แสดงดังตัวอย่างที่ 5.7

<u>ตัวอย่างที่ 5.7</u> แสดงแฟคทอเรียลฟังก์ชัน ในลักษณะของฟังก์ชันปกติ

```
#include <stdio.h>
long int fac ( int n) {
    long int result=1;
    int I;
    for (i = n; i > 0; i-- )
        result *= i;
    return ( result );
}
main ( ) {
   int n;
   printf (“Input n :> “);
   scanf(“%d” , &n);
   printf(“ %d factorial is :> %ld “, n, fac(n) );
}
```

<u>ตัวอย่างที่ 5.8</u> การหาค่าเลขยกกำลังด้วยฟังก์ชันแบบเรียกซ้ำ

```
#include <stdio.h>
float power (float val, int pow) {
    if (pow == 0)
        return(1.0);
    else
        return(power(val, pow-1) * val);
}
void main( ) {
    float b=10.0, ans;
    int p=4;
    ans = power ( b, p );
    printf("Answer of %f power %d is %f", b, p, ans);
}
```

5.6 ตัวอย่างเพิ่มเติม

ในหัวข้อเป็นอย่างเพิ่มเติมในเรื่องของฟังก์ชัน ซึ่งจะมีการแยกฟังก์ชันตามที่โจทย์ระบุเพื่อเป็นแนวทางในการเขียนโปรแกรม ทั้งนี้หากต้องการแยกฟังก์ชันเพิ่มมากกว่าในตัวอย่างก็สามารถทำ ได้ตามความต้องการและความเหมาะสม เสริมสร้างประสบการณ์ในการเขียนโปรแกรม ตัวอย่างเพิ่มเติมแสดงดังตัวอย่างที่ 5.9 ถึง 5.11

ตัวอย่างที่ 5.9 ให้เขียนโปรแกรมเพื่อรับข้อมูลจำนวนชั่วโมง นาที และวินาทีจากผู้ใช้ และให้คำนวณว่าเวลาทั้งหมดที่รับเข้ามานั้นเป็นกี่วินาที โดยให้เขียนส่วนของการคำนวณวินาทีเป็นฟังก์ชัน

```
#include <stdio.h>
#include <conio.h>
long convert(long h, long m, long s) {
    long total;
    total = (h * 60 + m) * 60 + s;
    return (total);
}
void main() {
    long hours, minutes, seconds;
    long time;
    clrscr();
    printf("Enter the time to be converted (hh mm ss) : ");
    scanf("%ld %ld %ld", &hours, &minutes, &seconds);
    time = convert(hours, minutes, seconds);
    printf("\nConvert to %ld seconds", time);
    getch();
}
```

ตัวอย่างที่ 5.10 เขียนโปรแกรมเพื่อแสดงรูปกรอบสี่เหลี่ยมโดยใช้เครื่องหมาย * ให้ผู้ใช้ระบุความกว้างและความสูงของรูปสี่เหลี่ยมที่ต้องการ

```
#include <stdio.h>
#include <conio.h>
void printHead(int width) {
    int i;
    for (i=0; i < width+2; i++)
        printf("*");
    printf("\n");
}
void printBody(int width, int height) {
    int i, j;
    for (i=0; i < height; i++) {
        printf("*");
        for (j=0; j < width; j++)
            printf(" ");
        printf("*\n");
    }
}
void main() {
    int w, h;
    clrscr();
    printf("Enter width : ");
    scanf("%d", &w);
    printf("Enter height : ");
    scanf("%d", &h);
    printf("\n\n");
    printHead(w);
    printBody(w, h);
    printHead(w);
    getch();
}
```

ตัวอย่างที่ 5.11 ให้เขียนโปรแกรมเพื่อรับข้อมูล จำนวนเต็ม 2 จำนวนจากผู้ใช้ โดยที่ข้อมูลนั้นต้องมากกว่า 20 และน้อยกว่า 60 และหาผลคูณของข้อมูลทั้ง 2

```
#include <stdio.h>
#include <conio.h>
#define YES 1
#define NO 0
int checkNumber(int, int);
int multiply(int, int);
void main() {
    int x, y, ans;
    int correct=NO;
    clrscr();
    do {
        printf("Enter x : ");
        scanf("%d", &x);
        printf("Enter y : ");
        scanf("%d", &y);
         correct = checkNumber(x, y);
    } while (!correct);
    ans = multiply(x, y);
    printf("\nMultiply of %d and %d is %d", x, y, ans);
    getch( );
}
```

```
int checkNumber(int a, int b) {
    if (a > 20 && a < 60 && b > 20 && b < 60)
        return (YES);
    else {
         printf("Input number in range > 20 and <60\n");
         return(NO);
    }
}
int multiply(int num1, int num2) {
    return(num1 * num2);
}
```

5.7 ไลบรารีมาตรฐานของภาษาซี

นอกเหนือจากฟังก์ชันต่าง ๆ ที่ผู้ใช้สามารถเขียนขึ้น ภาษาซีได้เตรียมฟังก์ชันมาตรฐานซึ่งจัดเก็บอยู่ในลักษณะที่เรียกว่าคลังโปรแกรมหรือไลบรารี (Library) ตัวอย่างของฟังก์ชันในไลบรารีมาตรฐานที่ได้มีการยกตัวอย่างไปแล้วคือ scanf( ) ที่ใช้ในการรับข้อมูล และ printf( ) ที่ใช้ในการแสดงผลข้อมูล และยังประกอบด้วยฟังก์ชันอื่น ๆ ที่คอมไพเลอร์ในภาษาซีของทุกบริษัทผู้ผลิตจะต้องมี และคอมไพเลอร์ของบางบริษัทอาจจะมีการจัดฟังก์ชันให้อยู่ในกลุ่มของอินคลูซไฟล์ที่ต่างกันซึ่งสามารถตรวจสอบได้รายละเอียดจากคู่มือของคอมไพเลอร์แต่ละบริษัท ในที่นี้จะยกตัวอย่างเฉพาะฟังก์ชันในไลบรารีมาตรฐานที่น่าสนใจดังนี้

5.7.1 ฟังก์ชันเกี่ยวกับตัวอักขระ

เมื่อต้องการใช้ฟังก์ชันเกี่ยวกับตัวอักขระ จะต้องมีการเรียกใช้อินคลูซไฟล์ <ctype.h> ด้วยคำสั่ง #include <ctype.h>

ฟังก์ชัน	หน้าที่
int isalnum(c)	ตรวจสอบตัวอักขระนั้นเป็นตัวอักษร 0-9 หรือ a-z หรือ A-Z หรือไม่ (Alphanumeric) ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0
int isalpha(c)	ตรวจสอบตัวอักขระนั้นเป็นตัวอักขระ a-z หรือ A-Z หรือไม่ ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0
int isdigit(c)	ตรวจสอบตัวอักขระนั้นเป็นตัวอักขระ 0-9 หรือไม่ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0
int islower(c)	ตรวจสอบตัวอักขระนั้นว่าเป็นตัวอักขระ a-z หรือไม่ ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0
int isupper(c)	ตรวจสอบตัวอักขระนั้นว่าเป็นตัวอักขระ A-Z หรือไม่ ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0
int tolower(c)	แปลงค่าอักขระปัจจุบันให้เป็นอักขระเดียวกันแต่เป็นตัวพิมพ์เล็ก
int toupper(c)	แปลงค่าอักขระปัจจุบันให้เป็นอักขระเดียวกันแต่เป็นตัวพิมพ์ใหญ่

ฟังก์ชัน	หน้าที่
int isalnum(c)	ตรวจสอบตัวอักขระนั้นเป็นตัวอักษร 0-9 หรือ a-z หรือ A-Z หรือไม่ (Alphanumeric) ถ้าไม่ใช่คืนค่า 0 ถ้าใช่คืนค่าที่ไม่ใช่ 0

5.7.2 ฟังก์ชันเกี่ยวกับตัวอักขระ

การเรียกใช้งานฟังก์ชันอรรถประโยชน์ทั่วไปจะต้องเรียกใช้อินคลูซไฟล์ <stdlib.h>

ฟังก์ชัน	หน้าที่
int abs(int n)	หาค่าสัมบูรณ์ของค่า n
float atof(const char *s)	แปลงค่าสตริง s เป็นค่าข้อมูลชนิด float
int atoi(const char *s)	แปลงค่าสตริง s เป็นค่าข้อมูลชนิด int
long atol(const char *s)	แปลงค่าสตริง s เป็นค่าข้อมูลชนิด long
void exit(int status)	เป็นคำสั่งเพื่อใช้ในการสิ้นสุดการทำงานของโปรแกรม โดยมีการคืนค่าสถานะของการทำงานของโปรแกรมให้กับระบบ โดยทั่วไปหากโปรแกรมทำงานสำเร็จไม่มีข้อผิดพลาดจะคืนค่า 0 แต่หากมีข้อผิดพลาดจะคืนค่าที่ไม่ใช่0 ซึ่งผู้เขียนโปรแกรมอาจจะกำหนดค่าต่าง ๆ ขึ้นให้สอดคล้องกับความผิดพลาดที่เกิด
void *malloc(size_t size)	ใช้จองพื้นที่ในหน่วยความจำ มีขนาดเท่ากับ size ไบต์ คืนค่าเป็นพอยน์เตอร์ชี้ไปยังพื้นที่ถ้าจองได้ แต่ถ้าไม่สามารถจองพื้นที่ได้จะคืนค่า NULL
int rand(void)	คืนค่าเลขจำนวนเต็มที่สุ่มขึ้นมาอยู่ในช่วง 0 ถึงRAND_MAX ซึ่งไม่เกิน 32767
void *realloc(void *ptr, size_t size)	ใช้จัดการพื้นที่ในหน่วยความจำ ที่มีการจองไว้ก่อนหน้านี้ด้วยคำสั่ง malloc( ) ในตัวแปร ptr ใหม่ ด้วยขนาด size ไบต์ ถ้าไม่สามารถจองพื้นที่ใหม่ได้จะคืนค่า NULL ปกติจะ เพื่อการเพิ่มหรือลดขนาดของพื้นที่ในหน่วยความจำ ที่ได้จองไว้ก่อนหน้านี้

5.7.3 ฟังก์ชันเกี่ยวข้องกับการรับและแสดงผลข้อมูล

การใช้งานฟังก์ชันเกี่ยวข้องกับการรับและแสดงผลข้อมูล จะต้องมีการเรียกใช้อินคลูซไฟล์ <stdio.h>

ฟังก์ชัน	หน้าที่
int getchar( )	อ่านข้อมูล 1 อักขระจากอุปกรณ์รับข้อมูลมาตรฐาน
char *gets(const char *buffer)	อ่านข้อมูลสตริงจากอุปกรณ์รับข้อมูลมาตรฐานมาเก็บในตัวแปร buffer จนกว่าจะมีการขึ้นบรรทัดใหม่หากมีความผิดพลาดจะคืนค่า NULL
int printf(const char *format, ...)	ใช้แสดงผลข้อมูลทางอุปกรณ์แสดงผลมาตรฐานในรูปแบบที่กำ หนด

int putchar(int c)	ส่งอักขระ c ไปแสดงผลทางอุปกรณ์แสดงผลมาตรฐาน
int puts(const char *buffer)	ส่งข้อมูลสตริง buffer ไปแสดงผลทางอุปกรณ์แสดงผลมาตรฐาน
int scanf(const char *format,...)	ใช้อ่านข้อมูลทางอปุกรณ์รับข้อมูลมาตรฐาน ในรูปแบบที่กำหนด
int sprintf(char *buffer, const char *format, ...)	ทำหน้าที่คล้ายกับ printf( ) แตจะส่งผลของข้อมูล ในรูปแบบที่กำหนดไปเก็บไว้ในสตริง buffer

5.7.4 ฟังก์ชันทางคณิตศาสตร์

การใช้ฟังก์ชันทางคณิตศาสตร์จะต้องเรียนอินคลูซไฟล์ <math.h>

ฟังก์ชัน	หน้าที่
acos(x)	หาค่า Arc Cosine ของ x
asin(x)	หาค่า Arc Sine ของ x
atan(x)	หาค่า Arc Tangent ของ x
atan2(x,y)	หาค่า Arc Tangent ของ x/y
ceil(x)	หาค่าจำนวนจริง x ที่มีการปัดเศษทิ้งทั้งหมด
cos(x)	หาค่า Cosine ของ x
exp(x)	หาค่าเอ็กโปเนนเชียล (Exponential) ของ x
floor(x)	หาค่าจำนวนจริง x ที่มีการปัดเศษขึ้นทั้งหมด
log(x)	หาค่า Logarithm ของ x
log10(x)	หาค่า Logarithm ฐาน 10 ของ x
pow(x,y)	หาค่า x ยกกำลัง y
sin(x)	หาค่า Sine ของ x
sqrt(x)	หาค่า Square Root ของ x

ตัวอย่างที่ 5.12 ให้รับข้อมูลเลขจำนวนจริงจากผู้ใช้ และแปลงค่าเป็นเลขจำนวนเต็ม โดยให้มีการปัดเศษ หากเศษมีค่ามากกว่า .50 ให้ปัดเศษขึ้น แต่ถ้าน้อยกว่าให้ปัดเศษลง

```
#include <stdio.h>
#include <conio.h>
void main( ) {
    float f;
    char s[21];
    int i;
    clrscr();
    printf("Enter number : ");
    scanf("%f", &f);
    sprintf(s, "%.0f", f);
    i = atoi(s);
    printf("Integer number is %d", i);
    getch( );
}
```