การทดลองที่ 5 ฟังก์ชัน (Function)

วัตถุประสงค์

1. ศึกษาการทำงานของฟังก์ชัน (Function)
2. ฝึกการเรียกใช้งานฟังก์ชันมาตรฐาน และฟังก์ชันที่สร้างขึ้นเองได้ถูกต้อง
3. ศึกษาฟังก์ชันที่มีการรับส่งค่าแบบต่างๆกัน

การทดลองตอนที่ 1 ศึกษาการทำงานและการเรียกใช้งานฟังก์ชัน

1. สังเกตลักษณะการทำงานของโปรแกรมแบบมีฟังก์ชัน โดย debug แบบ Step-Into ให้สังเกตว่าเมื่อมีการเรียกใช้งานฟังก์ชัน โปรแกรมจะกระโดดไปทำงานที่ฟังก์ชันที่เรียก และจะกลับมายังฟังก์ชัน main() หลังจากที่ทำงานที่ฟังก์ชันนั้นเรียบร้อยแล้ว โดยจะกลับมาที่คำสั่งถัดจากคำสั่งที่เรียกใช้งานฟังก์ชัน

   1.1 พิมพ์โปรแกรม lab5_1.c

```
1.  #include<stdio.h>
2.  void get_Fx(int x);
3.  void main()
4.  {
5.    int first, second;
6.     printf(“\n F(x) = 3x + 10 if X>0\n”);
7.     printf(“F(x) = 10 if X < 0 or X = 0”);
8.     printf(“\n\nEnter first value: ”);
9.     scanf(“%d”, &first);
10.    get_Fx(first);
11.    printf(“\n\nEnter second value: ”);
12.    scanf(“%d”, &second);
13.    get_Fx(second);
14.   system("PAUSE");
15. }
16. void get_Fx(int x)
17. {
18.    if (x>0)
19.      printf(“F(%d) is %d”, x, (3*x) + 10);
20.    else
21.      printf(“F(%d) is 10”, x);
22. }
```

   1.2 Set break point ไว้ที่บรรทัดที่ 5 แล้ว Run โปรแกรมแบบ Single-Step จนกระทั่งถึงบรรทัดที่ 10 สังเกตผลว่าเกิดอะไรขึ้นเมื่อผ่านบรรทัดที่ 10 พบว่า Compiler ไปทำงานคำสั่งที่บรรทัดใด ทำไมจึงไปที่บรรทัดดังกล่าว

   1.3 Run โปรแกรมแบบ Single-Step ไปเรื่อยๆ จนกระทั่งถึงบรรทัดที่ 11 หยุดสังเกตผล บันทึกว่าก่อนที่ Compiler จะมาทำคำสั่งที่บรรทัด 11 Compiler ทำคำสั่งที่บรรทัดใด และหลังจากทำงานที่บรรทัดดังกล่าวแล้วทำไมจึงไปทำงานที่บรรทัดที่ 11

1.4 Run โปรแกรมแบบ Single-Step จนกระทั่งถึงบรรทัดที่ 13 สังเกตผลว่าเกิดอะไรขึ้น เมื่อผ่านบรรทัดที่ 13 แล้ว Compiler ไปทำงานคำสั่งที่บรรทัดใด ทำไมจึงไปที่บรรทัดดังกล่าว เปรียบเทียบผลและคำสั่งในโปรแกรม กับผลและคำสั่งในการทดลองข้อที่ 1.2

2. ใช้งานฟังก์ชันมาตรฐานอย่างง่าย ๆ ได้ โดยดูจากต้นแบบฟังก์ชัน (Function prototype)

2.1 พิมพ์โปรแกรม lab5_2.c

```
1. #include<stdio.h>
2. #include<math.h>
3. main()
4. {
5.    double x;
6.    x = sin(39);
7.    printf("sin 39 is %f\n", x);
8.   printf("sin 85 is %f\n", sin(85));
9.   system("PAUSE");
10.}
```

2.2 ศึกษาต้นแบบฟังก์ชันของฟังก์ชัน sin ต่อไปนี้แล้ว สังเกตคำสั่งการเรียกใช้งานฟังก์ชันของโปรแกรมในบรรทัดที่ 6 และ 8

```
#include <math.h>
double sin( double arg );
```

The function sin( ) returns the sine of *arg*, where *arg* is given in radians.

2.3 ทดลองเปลี่ยนค่าอาร์กิวเมนต์ที่ให้กับฟังก์ชัน sin สังเกตผลว่าได้ตามที่คาดไว้หรือไม่ (ให้สังเกตว่าค่าอาร์กิวเมนต์ที่ให้กับฟังก์ชัน sin เป็นค่า radian ไม่ใช่ค่าองศา)

Checkpoint 1 จากโปรโตไทป์ของฟังก์ชันมาตรฐานจาก math.h ต่อไปนี้ จงเขียนโปรแกรมเรียกใช้งานฟังก์ชันเหล่านี้ให้ถูกต้อง (ฟังก์ชันมาตรฐานสามารถเรียกใช้เลย ไม่ต้องประกาศโปรโตไทป์เอง)

```
#include <math.h>
double pow( double base, double exp );
double sqrt(double x) ;
```

ฟังก์ชัน pow(base, exp) ให้ค่าคืนกลับเป็น base ยกกำลัง exp ($base^{exp}$)

ฟังก์ชัน sqrt(x) ให้ค่าคืนกลับเป็นรากที่สอง(square root) ของ x หรือ $\sqrt{x}$ เมื่อ $x \geq 0$

เช่น เขียนโปรแกรมรับค่าจำนวนจริง a, b และ x จากผู้ใช้และแสดงผลลัพธ์ a^b และ $\sqrt{x}$

การทดลองตอนที่ 2 ศึกษาฟังก์ชันที่มีการรับส่งค่าแบบต่างๆ

3. ฟังก์ชันที่ไม่มีการรับค่าและไม่มีการส่งค่ากลับ

   3.1 พิมพ์โปรแกรม lab5_3.c

```
1.  #include<stdio.h>
2.  void myfunc(void);
3.  void main()
4.  {
5.    printf("Hello, from main()\n");
6.    myfunc();
7.    printf("Hello, after calling myfunc()\n");
8.    system("PAUSE");
9.  }
10.  void myfunc()
11. {
12.   printf("Hello, from myfunc()\n");
13. }
```

   3.2 สังเกตผลการทำงานของโปรแกรม ดูลำดับการพิมพ์ข้อความ เมื่อมีการเรียกใช้งานฟังก์ชัน

4. ฟังก์ชันที่มีการรับค่าเข้าและมีการส่งค่ากลับ

   5.1 พิมพ์โปรแกรม lab5_4.c

   5.2 ทำการ Add Watch ตัวแปร a, b, x1 และ x2 จากนั้น Run โปรแกรมแบบ Single-Step โดยสังเกตค่าตัวแปร x1, x2 ที่บรรทัดที่ 8 และตัวแปร a, b หลัง Compile ผ่านบรรทัดที่ 8 เปรียบเทียบค่าของ x1, x2 และ a, b สังเกตว่าค่าของตัวแปร a และตัวแปร b ในฟังก์ชัน addvalue มาจากไหน

   5.3 สังเกตค่าตัวแปร a, b หลัง Compile ผ่านบรรทัดที่ 9 และค่าตัวแปร x1 ที่บรรทัดที่ 10 เปรียบเทียบค่าของ x1 และ a+b

โปรแกรม lab5_4.c

```
1.  #include<stdio.h>
2.  int addvalue(int a, int b);
3.  void main(void)
4.  {
5.    int x1, x2;
6.    printf("Enter x1 :"); scanf("%d", &x1);
7.    printf("Enter x2 :"); scanf("%d", &x2);
8.    printf("x1 + x2 = %d\n", addvalue(x1, x2));
9.    x1 = addvalue(5, 3);
10.   printf("5 + 3 = %d\n", x1);
11.   x1 = addvalue(-3, addvalue(5, addvalue
12.           (addvalue(12,4), -23)));
13.   printf("-3 + 5 + 12 + 4 – 23 = %d\n", x1);
14    System("PAUSE");
15. }
16. int addvalue(int a, int b)
17. {
18.   if (a > 32000)
19        return 0;
20    if (b > 32000)
21        return 0;
22    return a+b;
23 }
```

Checkpoint 2 จากตัวอย่างโปรแกรม lab5_4.c จงเขียนโปรแกรมหาค่าพื้นที่ของสี่เหลี่ยมผืนผ้า (rectangle) โดยรับค่าด้านกว้างและด้านยาวจากผู้ใช้ กำหนดให้ส่วนที่ใช้ในการคำนวณค่าพื้นที่ของสี่เหลี่ยมอยู่ในฟังก์ชัน Area

งานท้ายการทดลอง

1. จงเขียนโปรแกรมหาค่า f(x) โดยสมการ f(x) เป็นดังนี้

$$f(x) = x^2 + 2x + 3 \quad \text{if } x < 0$$
$$= 0 \quad \text{if } x = 0$$
$$= x - 2 \quad \text{if } x > 0$$

กำหนดให้ส่วนที่ใช้ในการคำนวณค่า f(x) อยู่ในฟังก์ชัน Get_Fx กำหนดให้ส่วนที่รับค่าตัวแปร X จากคีย์บอร์ด และส่วนที่แสดงผลลัพธ์ของค่า f(x) อยู่ในฟังก์ชัน main ห้ามใช้ตัวแปร Global ในโปรแกรมเด็ดขาด

2. จงเขียนโปรแกรมหาค่า f(x,y) โดยสมการ f(x,y) เป็นดังนี้

f(x, y) = x + y if x + y > 0

= 0 if x + y = 0

= -x + y if x + y < 0

กำหนดให้ส่วนที่ใช้ในการคำนวณค่า f(x,y) อยู่ในฟังก์ชัน Get_Fxy กำหนดให้ส่วนที่รับค่าตัวแปร x จากคีย์บอร์ด และส่วนที่แสดงผลลัพธ์ของค่า f(x,y) อยู่ในฟังก์ชัน main ห้ามใช้ตัวแปร Global ในโปรแกรมเด็ดขาด

3. จงเขียนโปรแกรมคำนวณปริมาตรรูปทรงกระบอก (cylinder) โดยรับค่าความยาวเส้นผ่านศูนย์กลาง (d, diameter) และ ความสูง (h, height) จากผู้ใช้ ให้สร้างฟังก์ชัน cylinder สำหรับการคำนวณดังกล่าว กำหนดให้ PI=3.14159 และสูตรการหาปริมาตรคือ $v = \pi(\frac{d}{2})^2 h$
