วิชา การพัฒนาโปรแกรม (ภาษาซี)

รหัสวิชา ว31293

หน่วยการเรียนรู้ที่ 8 โปรแกรมโมดูล

สาระการเรียนรู้

1. รู้จักกับฟังก์ชันภาษาซี
2. รูปแบบการเข้าถึงฟังก์ชัน
3. ฟังก์ชันต้นแบบ (Protptype)

ผลการเรียนรู้

ออกแบบโปรแกรมแบบโมดูลในรูปแบบของฟังก์ชันย่อย และสามารถเลือกใช้งานตัวแปรพอยน์เตอร์เพื่อประยุกต์ใช้งานในโปรแกรมได้อย่างถูกต้องเหมาะสม

จุดประสงค์การเรียนรู้

1. เข้าใจหลักการอาร์เรย์ และสามารถนำไปประยุกต์ใช้งานได้อย่างถูกต้อง
2. เปรียบเทียบความแตกต่างระหว่างอาร์เรย์ 1 มิติ และอาร์เรย์ 2 มิติได้
3. สามารถเขียนโปรแกรมเพื่อจัดเก็บข้อมูลลงในตัวแปรอาร์เรย์ได้
4. สามารถเขียนโปรแกรมเพื่อจัดการกับข้อความหรือสตริงได้
5. สามารถแปลงข้อความที่เป็นตัวเลข นำไปใช้เพื่อการคำนวณได้

จัดทำโดย นางพรพนารัตน์ ชมภูนุช ตำแหน่ง ครู

ใบความรู้ที่ 8 เรื่อง โปรแกรมโมดูล

จัดทำโดย นางพรพนารัตน์ ชมภูนุช

จากความรู้หลักการเขียนโปรแกรมในบทที่ 1 ที่มีการกล่าวว่า ควรออกแบบโปรแกรมเป็นส่วน ๆ หรือที่เรียกว่า โมดูล เพื่อให้แต่ละโมดูลทำงานในส่วนเฉพาะส่วนนั้น ๆ ซึ่งสิ่งเหล่านี้จะช่วยลดความซับซ้อนของโปรแกรมลงได้ อีกทั้งยังช่วยให้การตรวจสอบ และแก้ไขโปรแกรมทำได้ง่ายขึ้น และจากตัวอย่างโปรแกรมที่ผ่านมา ไม่ได้มีการแตกกระบวนการออกเป็นโมดูลย่อยเลย ซึ่งจะพบชุดคำสั่งทำงานทั้งหมด จะอยู่ภายในฟังก์ชัน main() ทั้งสิ้น กระบวนการทำงานของโปรแกรม ล้วนรวมอยู่ที่เดียวกัน และหากมีกระบวนการที่ต้องทำซ้ำ แทนที่จะเรียกใช้งานส่วนที่ซ้ำได้ กลับต้องเขียนโปรแกรมส่วนนั้นซ้ำอีก ส่งผลให้เกิดความซ้ำซ้อน และนับว่าเป็นแนวทางที่ไม่ถูกต้องนัก

ฟังก์ชันจึงเป็นส่วนของโปรแกรมที่มีความสมบูรณ์ภายในตัวเอง ในภาษาซีจะประกอบด้วยฟังก์ชันหนึ่งฟังก์ชันหรือหลายฟังก์ชัน โดยในแต่ละโปรแกรมจะต้องมีอย่างน้อย 1 ฟังก์ชัน คือ ฟังก์ชัน main() ส่วนฟังก์ชันอื่น ๆ จะเป็นส่วนย่อยที่อยู่ในฟังก์ชัน main() หรืออาจเป็นส่วนย่อยของฟังก์ชันอื่น ๆ ก็ได้ ฟังก์ชันในภาษาซีจะมีลักษณะคล้ายโปรแกรมย่อย หรือที่เรียกว่า **สับรูทีน (Subrutine Funchtion)**

ฟังก์ชันเป็นองค์ประกอบพื้นฐานของโปรแกรมในภาษาซี การทำงานต่าง ๆ ของโปรแกรมจะต้องเขียนอยู่ในรูปฟังก์ชัน โดยที่ฟังก์ชัน main() ซึ่งเป็นฟังก์ชันแรกของโปรแกรมภาษาซีที่จะถูกเรียกให้ทำงาน และนักเขียนโปรแกรมก็สามารถเขียนฟังก์ชันที่มีขนาดเล็กเพื่อให้ทำงานเฉพาะเจาะจงบางอย่าง และยังสามารถนำฟังก์ชันเหล่านี้มาใช้ได้อีกในภายหลัง ความสามารถของภาษาซีเหล่านี้จึงทำให้เกิดความสะดวกในการทำโปรแกรมขนาดใหญ่ และช่วยให้การตรวจสอบความถูกต้องของโปรแกรม หรือการแก้ไขโปรแกรมในภายหลังทำงานได้ง่ายขึ้น ดังนั้น ในการเขียนโปรแกรมภาษาซีที่ดีจึงจำเป็นต้องออกแบบโปรแกรมให้เป็นโมดูลหรือที่เรียกว่าฟังก์ชันย่อย ๆ ก็เพราะว่า

1. เพื่อเป็นไปตามหลักการเขียนโปรแกรมเชิงโครงสร้าง
2. เพื่อง่ายต่อการตรวจสอบและการบำรุงรักษา
3. เพื่อหลีกเลี่ยงการเขียนชุดคำสั่งเดิม ที่ทำงานซ้ำ ๆ
4. เพื่อสร้างกลุ่มคำสั่งประมวลผลเฉพาะงาน

ดังนั้นในบทนี้ จึงมีจุดประสงค์เพื่อฝึกแนวคิดการเขียนโปรแกรมแบบใหม่ ด้วยการแตกการทำงานออกเป็นโมดูล ซึ่งในภาษาซีเรียกว่า **ฟังก์ชัน (Funchtion)** ซึ่งโปรแกรมเมอร์จะต้องจะต้องสร้างขึ้นเอง โดยรูปแบบการเข้าถึงฟังก์ชันก็จะมีหลายวิธีด้วยกัน อีกทั้งภาษาซียังอนุญาตให้สร้างฟังก์ชันเพื่อใช้งานได้โดยไม่จำกัด

รู้จักกับฟังก์ชันภาษาซี

ฟังก์ชันภาษาซี สามารถแบ่งได้เป็น 2 ประเภท คือ

- ฟังก์ชันมาตรฐาน หรือ ไลบรารีฟังก์ชัน (Library Funchtion)
- ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (User defined funchtion)

1. ฟังก์ชันมาตรฐาน (standard functions)

ฟังก์ชันมาตรฐานในภาษาซี ซึ่งจะอยู่ในไลบรารีภาษาซีมาตรฐาน (*C Standard Library*) ไลบรารีภาษาซีมาตรฐานประกอบด้วยฟังก์ชันต่างๆมากมาย ไม่ว่าจะใช้สำหรับการคำนวณทางคณิตศาสตร์ การจัดการกับข้อความ การจัดการกับ input/output และอื่นๆ ซึ่งจะทำให้งานของโปรแกรมเมอร์ง่ายขึ้น โดยการใช้งานฟังก์ชันประเภทนี้จะต้องรวม (*include*) ไลบรารีที่ต้องการใช้งานเพื่อให้ตัวแปลภาษารู้ว่าฟังก์ชันที่โปรแกรมเมอร์ต้องการใช้อยู่ในไลบรารีมาตรฐานตัวใด ตัวอย่างเช่น หากต้องการใช้ฟังก์ชัน printf() หรือ scanf() ซึ่งอยู่ในไลบรารีมาตรฐานสำหรับเกี่ยวกับอินพุตและเอาต์พุต *(standard input/output)* ที่ชื่อ *stdio.h*

1.1 ฟังก์ชันทางคณิตศาสตร์ (mathematic functions)

เป็นฟังก์ชันที่ใช้สำหรับการคำนวณทางคณิตศาสตร์ และก่อนที่จะใช้ฟังก์ชันประเภทนี้ จะต้องใช้คำสั่ง #include <math.h> แทรกอยู่ตอนต้นของโปรแกรม และตัวแปรที่จะใช้ฟังก์ชันประเภทนี้จะต้องมีชนิด (type) เป็น double เนื่องจากผลลัพธ์ที่ได้จากฟังก์ชันประเภทนี้จะได้ค่าส่งกลับของข้อมูลเป็น double เช่นกัน ฟังก์ชันทางคณิตศาสตร์ที่สำคัญ เช่น

ฟังก์ชัน acos(x) เป็นฟังก์ชันที่ใช้คำนวณหาค่า arc cosine ของ x โดยที่ x เป็นค่ามุมในหน่วยเรเดียน (radian)

ฟังก์ชัน exp(x) เป็นฟังก์ชันที่ใช้หาค่า ex โดยที่ x เป็นค่าคงที่หรือตัวแปรที่จะใช้เป็นค่ายกกำลังของ e โดยที่ e มีค่าประมาณ 2.718282

ฟังก์ชัน log10(x) เป็นฟังก์ชันที่ใช้หาค่า log ฐาน 10 ของค่าคงที่หรือตัวแปร x โดยที่ x เป็นค่าคงที่หรือตัวแปรที่มีค่าเป็นลบไม่ได้

1.2 ฟังก์ชันเกี่ยวกับตัวอักษร (character functions)

เป็นฟังก์ชันที่ใช้กับข้อมูลที่มีชนิดเป็น single char (ใช้เนื้อที่ 1 byte) เท่านั้น และก่อนที่จะใช้ฟังก์ชันประเภทนี้จะต้องใช้คำสั่ง #include <ctype.h> แทรกอยู่ตอนต้นของโปรแกรม จึงจะสามารถเรียกใช้ฟังก์ชันประเภทนี้ได้ ฟังก์ชันเกี่ยวกับตัวอักษรที่สำคัญ เช่น isalnum(ch), isalpha(ch), isdigit(ch) , islower(ch) , isupper(ch) , tolower(ch) , toupper(ch) , isspace(ch) , isxdigit(ch)

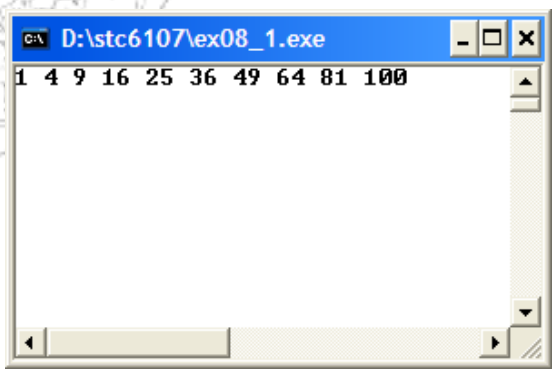
1.3 ฟังก์ชันเกี่ยวกับสตริง (string functions)

เป็นฟังก์ชันที่ใช้กับข้อมูลชนิดสตริง (string) โดยก่อนที่จะใช้ฟังก์ชันประเภทนี้จะต้องใช้คำสั่ง #include<string.h> แทรกอยู่ตอนต้นของโปรแกรมเสียก่อน จึงจะเรียกใช้ฟังก์ชันประเภทนี้ได้ ฟังก์ชันเกี่ยวกับสตริงที่สำคัญ เช่น strlen(s), strcmp(s1,s2), strcpy(s), strcat(s1,s2)

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมที่มีโครงสร้างโปรแกรมแบบมีฟังก์ชัน ในการแสดงข้อความออกทางหน้าจอ ให้บันทึกไฟล์ชื่อ ex08_1.cpp

```
1  #include <stdio.h>
2  #include<conio.h>
3  main()
4  {
5  int x;
6  for (x = 1; x <= 10; x++)
7  printf("%d ", x * x);
8  printf("\n");
9  return 0;
10 }
```

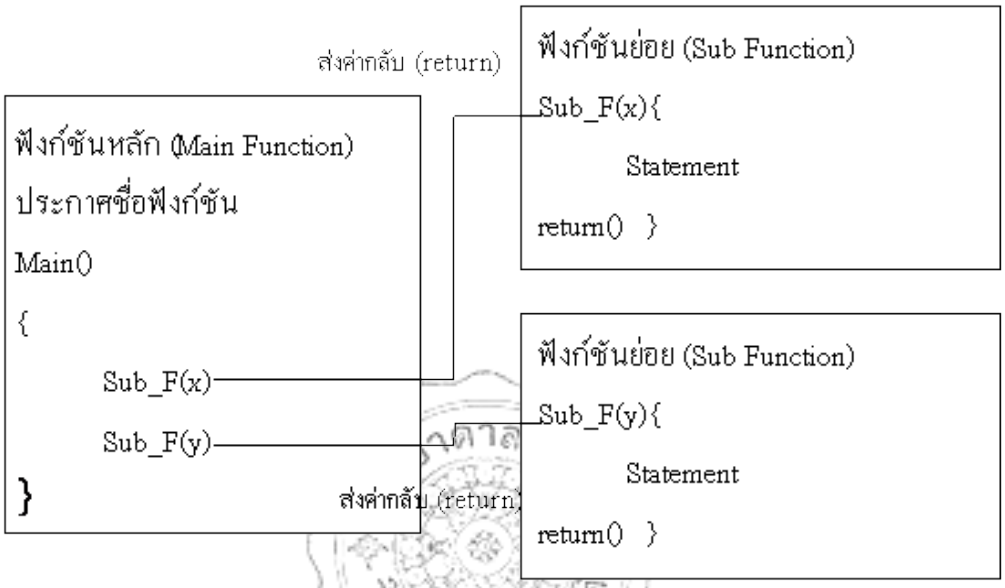


รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex08_1.exe

2. ฟังก์ชันที่เขียนขึ้นเอง (user defined functions)

ฟังก์ชันที่เขียนขึ้นหรือบางครั้งเราเรียกว่าโปรแกรมย่อย คือ ส่วนของโปรแกรมที่เขียนขึ้นเพื่อให้สามารถทำงานได้อย่างใดอย่างหนึ่งตามความต้องการ สำหรับผู้เขียนโปรแกรมภาษา C นิยมเรียกโปรแกรมย่อยว่า "ฟังก์ชัน" ส่วนผู้ที่เขียนโปรแกรมภาษาปาสคาล โคบอล ฟอร์แทรน เบสิก นิยมเรียกว่า "โปรแกรมย่อย" อย่างไรก็ตามโปรแกรมย่อยหรือฟังก์ชันก็มีลักษณะการทำงานเหมือนกัน

เพียงแต่เรียกชื่อต่างกันเท่านั้นสำหรับในเอกสารนี้จะเรียกว่า ฟังก์ชัน ซึ่งในเนื้อหาส่วนนี้จะเป็นการเรียนรู้เกี่ยวกับฟังชันที่เขียนขึ้น (user define functions : UDF) ตั้งแต่การประกาศรูปแบบฟังก์ชัน การเขียนตัวฟังก์ชันและการเรียกใช้ฟังก์ชันเพื่อให้สามารถใช้ฟังก์ชันที่เขียนขึ้นในโปรแกรมภาษา C ได้อย่างถูกต้อง



รูปที่ 7.1 แสดงโครงสร้างการส่งค่าของฟังก์ชัน

รูปแบบการเข้าถึงฟังก์ชัน

ในการเขียนฟังก์ชันขึ้นมาใช้งานอย่างใดอย่างหนึ่ง เราสามารถจำแนกฟังก์ชันที่เขียนขึ้นตามลักษณะการส่งค่าไปและรับค่ากลับได้ 4 แบบ คือ

1. ฟังก์ชันที่ไม่มีการส่งค่าไปและรับค่ากลับ
2. ฟังก์ชันที่มีการส่งค่าไปแต่ไม่มีรับค่ากลับ
3. ฟังก์ชันที่มีทั้งการส่งค่าไปและรับค่ากลับ
4. ฟังก์ชันที่ไม่มีการส่งค่าไปแต่มีการส่งค่ากลับ

ซึ่งฟังก์ชันแต่ละแบบก็เหมาะกับงานแต่ละอย่าง ดังนั้นผู้เขียนฟังก์ชันจึงจำเป็นที่จะต้องศึกษาทำความเข้าใจฟังก์ชันแต่ละแบบ เพื่อจะได้มาประยุกต์ใช้กับงานได้อย่างเหมาะสม และการสร้างฟังก์ชันขึ้นมาใช้งาน ต้องเขียนคำสั่งให้ถูกต้องตามรูปแบบของภาษาซี กำหนดไว้

type เป็นประเภทของฟังก์ชันที่ต้องการประกาศ ซึ่งขึ้นกับค่าผลลัพธ์ของฟังก์ชันเป็นอะไร คุณสามารถประกาศ function type ได้เหมือนกับ primitive datatype เช่น int, float, char และอื่นๆ สำหรับฟังก์ชันที่ไม่มีการส่งค่ากลับจะมี type เป็น void

name เป็นชื่อของฟังก์ชันที่คุณต้องการสร้าง ในการตั้งชื่อฟังก์ชันมันมีกฎเช่นเดียวกันเหมือนกับการตั้งชื่อตัวแปร ชื่อของฟังก์ชันเป็นสิ่งที่เราจะใช้เมื่อต้องการใช้งานฟังก์ชัน

argument เป็นตัวแปรที่ส่งเข้ามาในฟังก์ชัน พารามิเตอร์เป็นทางเลือกซึ่งสามารถมีหรือไม่ก็ได้

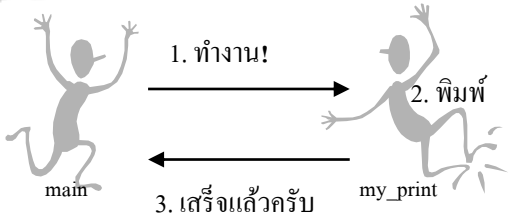
statement เป็นคำสั่งของโปรแกรมเพื่อที่จะให้ฟังก์ชันทำงานและได้ผลลัพธ์ที่ต้องการ

1. ฟังก์ชันที่ไม่มีการส่งผ่านค่าไปและรับค่ากลับ

จัดเป็นฟังก์ชันที่มุ่งเน้นให้ทำงานใด ๆ จนสำเร็จเท่านั้น โดยไม่มีการส่งค่าผ่านค่าใด ๆ ลงไป และไม่มีการคืนค่าใด ๆ กลับไปยังฟังก์ชัน ทั้งนี้ภาษาซีจะใช้คำว่า void โดยให้ระบุลงไปตรงส่วนชนิดข้อมูลที่คืนค่า และระบุ void ลงไปในอากิวเมนต์ที่อยู่เครื่องหมายวงเล็บ โดยคำว่า void เป็นตัวแจ้งให้คอมไพล์เลอร์ทราบว่า ฟังก์ชันนี้จะไม่มีการคืนค่า หรือไม่มีการส่งผ่านข้อมูลใด ๆ ทั้งสิ้น

รูปแบบ

```
void  function (void)
{
    statement_1;
    statement_2;
        ...
    statement_n;
}
```



ตัวอย่างที่ 1

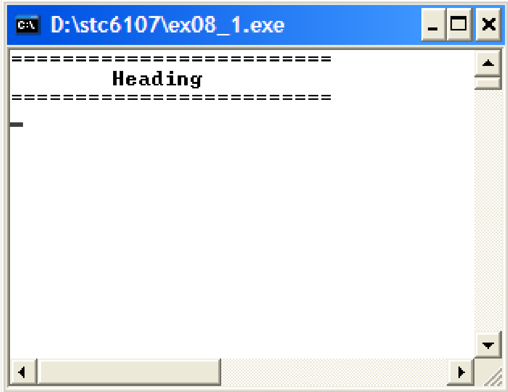
ฟังก์ชันที่ไม่รับผ่านค่าตัวแปร และไม่ส่งผ่านค่ากลับ

```
void   main()
{
    my_print();
}

void  my_print()
{
    printf("Hello world");
}
```

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมที่มีโครงสร้างโปรแกรมแบบมีฟังก์ชัน ในการแสดงข้อความออกทางหน้าจอ ให้บันทึกไฟล์ชื่อ ex08_2.cpp

```
#include <stdio.h>
#include<conio.h>
void  prtline(void)
{
puts("=========================");
}

main()
{
   prtline();
   printf("\t Heading \n");
   prtline();
   getche();
 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex08_2.exe

2. ฟังก์ชันที่มีการส่งผ่านค่าทางเดียว

เป็นฟังก์ชันที่มีการส่งผ่านค่าผ่านอาร์กิวเมนต์ และภายหลังเสร็จสิ้นการทำงานของตัวฟังก์ชันแล้ว จะไม่มีการคืนค่าใด ๆ กลับไป ดังนั้นอาร์กิวเมนต์ภายในวงเล็บ จึงต้องระบุชนิดข้อมูลลงไป ซึ่งจะมีกี่ตัวก็ได้ ในขณะที่ชนิดข้อมูลที่คืนค่ากลับ ให้ระบุเป็น void

รูปแบบ

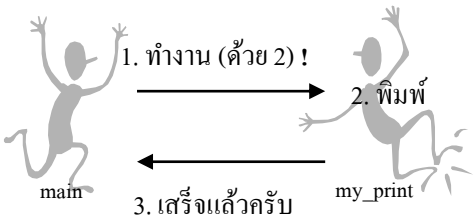
```
void  function (type  arg1 , type   arg2 , …)
 {
     statement_1;
     statement_2;
          …
     statement_n;
 }
```

ตัวอย่าง

กรณีที่ 1 ฟังก์ชันที่มีการรับผ่านค่าตัวแปร แต่ไม่ส่งผ่านค่ากลับ

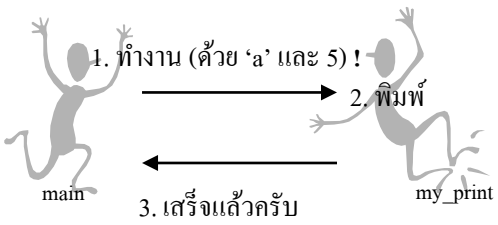
รูปแบบที่ 1.1

```
void main()
{
      my_print(2);
}

void my_print(int x)
{
      printf("%d", x);
}
```



รูปแบบที่ 1.2

```
void main()
{
      my_print('a', 5);
}

void my_print(char ch, int x)
{
      while (x > 0)
      {
      printf("%c", ch);
      x--;
      }
}
```



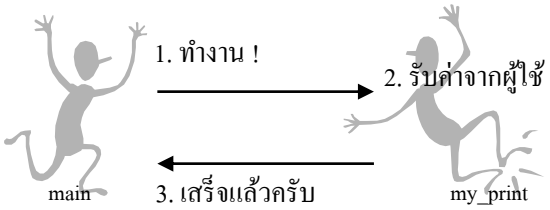
กรณีที่ 2 ฟังก์ชันที่ไม่มีการส่งค่าไปแต่มีการส่งค่ากลับ

เป็นรูปแบบฟังก์ชันที่ไม่มีการส่งผ่านค่าไปยังฟังก์ชัน ยังมีการคืนค่ากลับไปด้วยพิจารณาจากตัวอย่างจะพบว่า ฟังก์ชัน add() จะมีอาร์กิวเมนต์อยู่ 2 ตัวแปรและมีการคืนค่ากลับด้วยตัวเลขจำนวนเต็ม

ตัวอย่างที่ 4

ฟังก์ชันที่ไม่มีการรับผ่านค่า และแต่มีการส่งผ่านค่ากลับ

```
void main()
{
    char ch;
    ch = my_print();
    printf("%c\n", ch);
}

char my_print(void)
{
    char lch;
    printf("Enter your character: ");
    scanf("%c", &lch);

    printf("\n");
    return (lch);
}
```

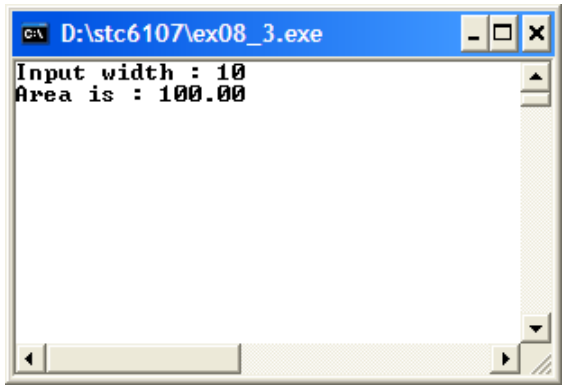


ผลคือ lch

กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมที่มีโครงสร้างโปรแกรมแบบมีฟังก์ชัน ในการแสดงข้อความออกทางหน้าจอ ให้บันทึกไฟล์ชื่อ ex08_3.cpp

```
1  #include <stdio.h>
2  #include<conio.h>
3  void square (int);
4  main()
5  {
6  int  w;
7  printf("Input width : ");
8  scanf("%d",&w);
9  square(w);
10 return 0;
11 }
12 void square (int  y)
13 {
14 float  ans;
15 ans=y*y;
16 printf("Area is : %.2f ", ans );
17 getche();
18 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex08_3.exe

3. ฟังก์ชันที่ส่งผ่านค่าและคืนค่ากลับ

เป็นรูปแบบฟังก์ชันที่นอกจากจะมีการส่งผ่านค่าแล้ว ยังมีการคืนค่ากลับไปด้วยพิจารณาจากตัวอย่างจะพบว่า ฟังก์ชัน add() จะมีอาร์กิวเมนต์อยู่ 2 ตัวแปรและมีการคืนค่ากลับด้วยตัวเลขจำนวนเต็ม

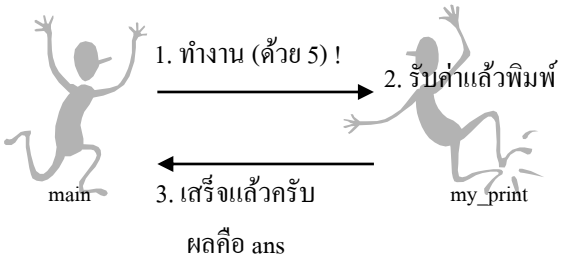
รูปแบบ

```
data_type   function (type  arg1,type   arg2,…)
 {
    statement_1;
    statement_2;
        ...
    statement_n;
    return(expression);
}
```

ตัวอย่างที่ 3

ฟังก์ชันที่มีทั้งการรับผ่านค่า และส่งผ่านค่ากลับ

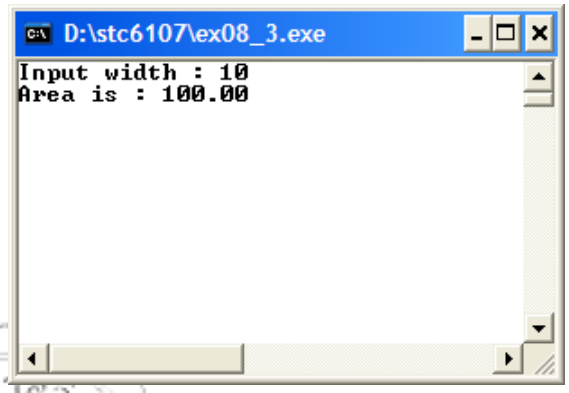
```
void main()
{
      char ch;
      ch = my_print(5);
      printf("%c\n", ch);
}

char my_print(int x)
{
      char  ans;
      printf("Enter your character: ");
      scanf("%c", &ans);
      while (x > 0)
      {
         printf("%c", ans);
          x--;
      }
      printf("\n");
      return ans;
}
```



กิจกรรมเสริมทักษะ

ให้นักเรียนทดลองเขียนโปรแกรมที่มีโครงสร้างโปรแกรมแบบมีฟังก์ชัน ในการแสดงข้อความออกทางหน้าจอ ให้บันทึกไฟล์ชื่อ ex08_4.cpp

```
1  #include <stdio.h>
2  #include <conio.h>
3  int  square(int);
4  main()
5  {
6  int x;
7  for (x = 1; x <= 10; x++)
8  printf("%d ", square(x));
9  printf("\n");
10 return 0;
11 }
12
13 int square(int y)
14 {
15       return y*y;
16 }
```



รูปภาพ แสดงผลลัพธ์ที่ได้จากการทำงานของโปรแกรม ex08_4.exe

ฟังก์ชันต้นแบบ (Protptype)

จากโปรแกรมการสร้างฟังก์ชันใช้งานเอง จะพบว่าได้มีการวางตำแหน่งฟังก์ชันที่สร้างขึ้นไว้ก่อนฟังก์ชัน main() ทั้งนี้ก็เพื่อให้คอมไพเลอร์ได้แปลชุดคำสั่งของฟังก์ชันที่สร้างขึ้นเองก่อน แต่อย่างไรก็ตามกรณีต้องการวางตำแหน่งฟังก์ชัน main() ไว้ก่อนฟังก์ชันอื่น ๆ ที่สร้างขึ้นเอง จำเป็นต้องประกาศฟังก์ชันต้นแบบไว้ที่ส่วนหัวของโปรแกรม (ก่อนฟังก์ชัน main()) เพื่อให้คอมไพล์เลอร์ได้รู้จักกับฟังก์ชันเหล่านั้นก่อน

การใช้วิธีสร้างฟังก์ชันต้นแบบ (Function Prototype) ก็คือ ฟังก์ชันต้นแบบจะเป็นตัวบอกให้ตัวแปลภาษารู้ถึงชนิดของข้อมูลที่จะส่งค่ากลับ จำนวนของตัวแปรพารามิเตอร์ ที่ฟังก์ชันคาดหวังว่าจะได้รับ ชนิดของพารามิเตอร์แต่ละตัว และลำดับของพารามิเตอร์เหล่านั้น ตัวแปลภาษาสามารถที่จะนำข้อมูลเหล่านี้ในการตรวจสอบความถูกต้องของการเรียกใช้ฟังก์ชัน ลองคิดถึงปัญหาที่ตามมา หากตัว

แปลภาษายอมให้มีการเรียกใช้ฟังก์ชันที่มีพารามิเตอร์ไม่ครบ หรือสลับชนิดของพารามิเตอร์ เมื่อเกิดความผิดพลาดขณะรันโปรแกรม จะตรวจสอบหาความผิดพลาดได้ยาก การใช้ต้นแบบของฟังก์ชันสามารถหลีกเลี่ยงความไร้ประสิทธิภาพเหล่านี้ได้

ตามปกติถ้าต้องการให้ตัวแปรภาษา C แปลความหมายโปรแกรมได้อย่างถูกต้อง เราจะต้องเขียนฟังก์ชันอื่นๆ ไว้ก่อนฟังก์ชัน main() แต่การเขียนโปรโตไทป์จะทำให้เราสามารถย้ายตำแหน่งของฟังก์ชันไปไว้ที่ส่วนใดของโปรแกรมก็ได้ โดยไม่จำเป็นต้องเขียนไว้ก่อนฟังก์ชัน main() ซึ่งรูปแบบการเขียนโปรโตไทป์หรือฟังก์ชันต้นแบบ จะเหมือนกับการเขียนบรรทัดแรกของฟังก์ชัน ดังต่อไปนี้

รูปแบบ

```
data_type   function (type-1 ,type-2 , … , type-n);
```

ตัวอย่าง

```
#include <stdio.h>
#include <conio.h>
Int  maximum(int , int , int);
main()
{
  int a, b, c;
  printf("Enter three integers: ");
  scanf("%d  %d  %d", &a, &b, &c);
  printf("Maximum is: %d\n", maximum(a, b, c);
  return 0;
}

int  maximum(int  x , int  y, int  z)
{
  int max = x;
  if (y > max)
          max = y;
  if (z > max)
          max = z;
  return max;
}
```

ประกาศฟังก์ชันต้นแบบ (Function Prototype)

ประกาศตัวแปรที่ใช้ในการรับค่าในฟังก์ชันที่ชื่อว่า maximum