แผนการสอนประจำบทที่ 5

ฟังก์ชัน

หัวข้อสำคัญ

1. ความหมายของฟังก์ชัน ชนิดของฟังก์ชัน และประโยชน์ของฟังก์ชัน
2. การประกาศ การสร้าง และการเรียกใช้ฟังก์ชัน
3. การส่งผ่านค่าระหว่างฟังก์ชัน
4. การประกาศฟังก์ชันต้นแบบ
5. การเข้าถึงฟังก์ชัน

วัตถุประสงค์เชิงพฤติกรรม

1. ผู้เรียนสามารถอธิบายความหมายของฟังก์ชันได้
2. ผู้เรียนสามารถเขียนโปรแกรมประกาศฟังชันก์ สร้างฟังก์ชัน และเรียกใช้ฟังก์ชันได้

วิธีการสอนและกิจกรรมการเรียนการสอน

1. การบรรยาย
2. การทำแบบฝึกหัด

สื่อที่ใช้ประกอบการสอน

1. เอกสารประกอบการสอน
2. เครื่องคอมพิวเตอร์
3. เครื่องฉายภาพนิ่ง

การวัดและประเมินผล

1. สังเกตจากความสนใจของผู้เรียน
2. ประเมินจากการตอบคำถามของผู้เรียนและกิจกรรมในชั้นเรียน
3. การทำแบบฝึกหัดท้ายบท

บทที่ 5
ฟังก์ชัน

5.1.ความหมายของฟังก์ชัน

ชุดคำสั่งที่เขียนขึ้นมาเพื่อให้คอมพิวเตอร์ทำงานต่าง ๆ สามารถรับข้อมูล ประมวลผล และแสดงผลข้อมูล แต่ละฟังก์ชันจะทำงานเสร็จภายในตัวเอง สามารถเรียกใช้ฟังก์ชันได้บ่อยตามต้องการ ฟังก์ชันสามารถเรียกใช้ฟังก์ชันอื่นเป็นทอดต่อ ๆ กันได้

5.2.ชนิดของฟังก์ชัน

ฟังก์ชันมี 2 ชนิด ได้แก่ ฟังก์ชันมาตรฐาน (Standard library functions) และฟังก์ชันที่สร้างขึ้นมาใช้งานเอง (User defined functions)

5.3.1 ฟังก์ชันมาตรฐาน (Standard library functions)

ฟังก์ชันมาตรฐานของภาษาซี คือ ฟังก์ชันของภาษาซีที่อยู่ในไลบรารี่ (library function) ของภาษาซี สามารถเรียกใช้งานฟังก์ได้ทันที เช่น ฟังก์ชันทางคณิตศาสตร์ เช่น math.h ฟังก์ชันเกี่ยวกับสตริง เช่น ctype.h, string.h เป็นต้น การเรียกใช้งานให้ประกาศ #include ตามด้วยชื่อฟังก์ชันมาตรฐานที่ต้องการใช้งานภายในเครื่องหมาย <> เช่น #include < math.h>

ตัวอย่างฟังก์ชันมาตรฐานในภาษา C

1.1) ฟังก์ชันที่ประกาศในเฮดเดอร์ไฟล์ math.h

- double sin(double x) หาค่ามุม sin ของค่า x
- double cos(double x) หาค่ามุม cos ของค่า x
- double tan(double x) หาค่ามุม tan ของค่า x
- double sqrt(double x) หาค่ารากที่สองของค่า x
- double pow(double x, double y) หาค่า x ยกกำลังด้วยค่า y
- double celi(double x) ปัดเศษทศนิยมขึ้น
- double floor(double x) ปัดเศษทศนิยมลง

ตัวอย่างการใช้ฟังก์ชัน math.h

โปรแกรม

```
1.  #include <stdio.h>
2.  #include <math.h>
3.  #define PI 3.14159265
4.  main () {
5.     double val, radius;
6.     val = 30.00;
7.     radius = PI / 180.0;
8.     printf("The sin of %.2lf is %.2lf degrees\n", val, sin(radius*val));
9.     printf("The cos of %.2lf is %.2lf degrees\n", val, cos(radius*val));
10.    printf("The tan of %.2lf is %.2lf degrees\n", val, tan(radius*val));

11.    printf("The sqrt of 36 is %lf\n", sqrt(36));
12.    printf("The pow(2,3) is %lf\n", pow(2,3));
13.    printf("The ceil(30.55) is %lf\n", ceil(30.55));
14.    printf("The ceil(30.11) is %lf\n", ceil(30.11));
15.    printf("The floor(30.55) is %lf\n", floor(30.55));
16.    printf("The floor(30.11) is %lf\n", floor(30.11));
17. }
```

ผลลัพธ์

1.2) ฟังก์ชันที่ประกาศในเฮดเดอร์ไฟล์ ctype.h

- int islower(int c) ตรวจสอบตัวอักษรว่าเป็นตัวอักษรพิมพ์เล็กหรือไม่ ค่ามากกว่า 0 เป็นตัวอักษรพิมพ์เล็ก และค่าเท่ากับ 0 ไม่เป็นตัวอักษรพิมพ์เล็ก
- int isupper(int c) ตรวจสอบตัวอักษรว่าเป็นตัวอักษรพิมพ์ใหญ่หรือไม่ค่ามากกว่า 0 เป็นตัวอักษรพิมพ์ใหญ่ และค่าเท่ากับ 0 ไม่เป็นตัวอักษรพิมพ์ใหญ่
- int tolower(int c) แปลงตัวอักษรเป็นตัวพิมพ์เล็ก
- int toupper(int c) แปลงตัวอักษรเป็นตัวพิมพ์ใหญ่

ตัวอย่างการใช้ฟังก์ชัน ctype.h

โปรแกรม	ผลลัพธ์
1. #include <stdio.h> 2. #include <ctype.h> 3. main() { 4. printf("a is a lowercase ? : %d\n",islower('a')); 5. printf("A is A lowercase ? : %d\n\n",islower('A')); 6. printf("a is a uppercase ? : %d\n",isupper('a')); 7. printf("A is A uppercase ? : %d\n\n",isupper('A')); 8. printf("a to lowercase is %c\n",tolower('a')); 9. printf("A to lowercase is %c\n\n",tolower('A')); 10. printf("a to uppercase is %c\n",toupper('a')); 11. printf("A to uppercase is %c\n\n",toupper('A')); 12. }	

5.3.2 ฟังก์ชันที่สร้างขึ้นมาใช้งานเอง (User defined functions)

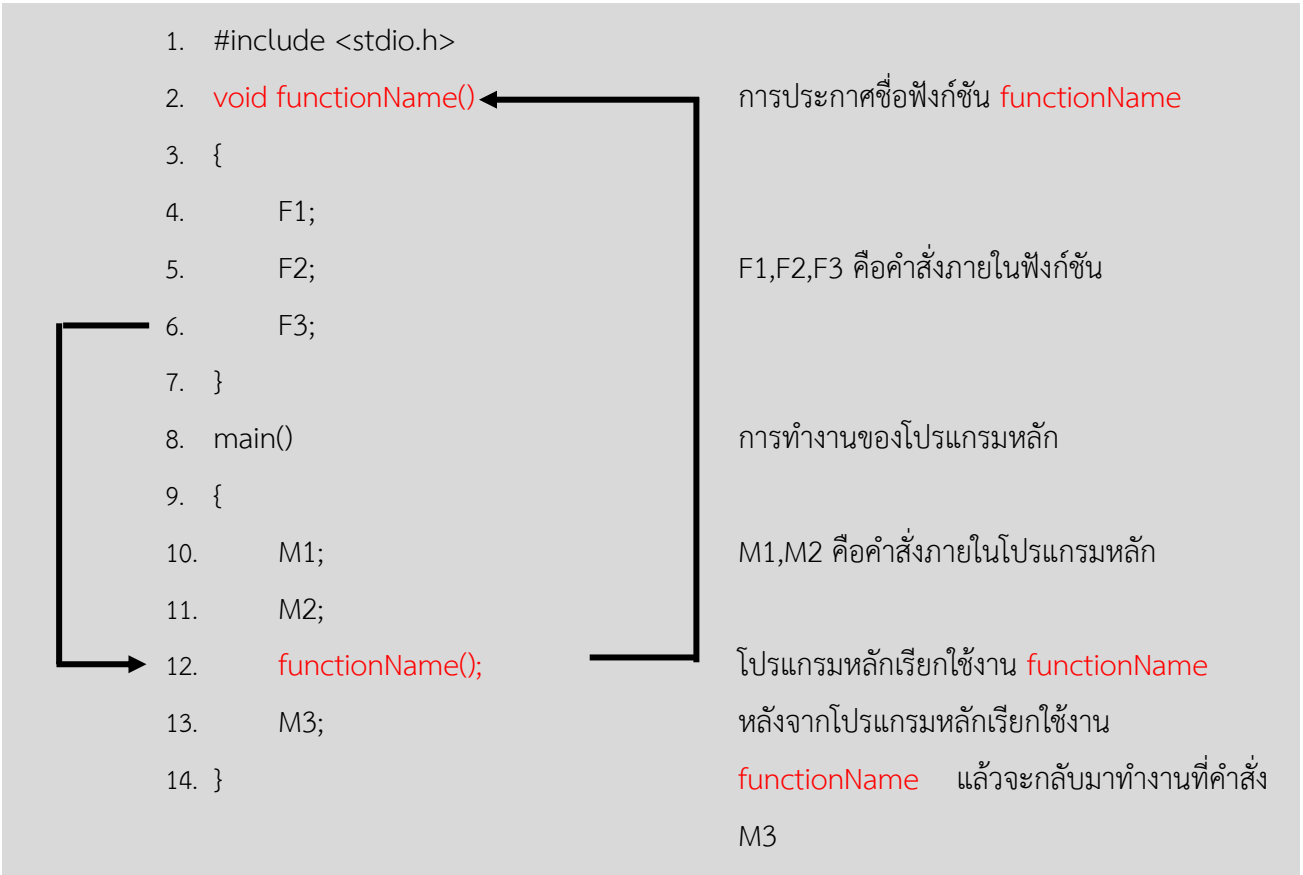
ฟังก์ชันที่สร้างขึ้นมาใช้งานเอง คือ ฟังก์ชันที่ผู้พัฒนาที่เขียนขึ้นมาเอง โดยมีจุดประสงค์เพื่อแก้ไขปัญหาอย่างใดอย่างขึ้น

5.3.ประโยชน์ของฟังก์ชัน

การแบ่งโปรแกรมเป็นฟังก์ชันทำให้สะดวกต่อการตรวจสอบและแก้ไข ทำให้โปรแกรมมีลักษณะเป็นโปรแกรมเชิงโครงสร้างเพื่อหลีกเลี่ยงการเขียนชุดคำสั่งเดิมซ้ำๆ ต้องการสร้างชุดคำสั่งที่ทำงานใดงานหนึ่งโดยเฉพาะ สามารถแก้ไขฟังก์ชันโดยไม่มีผลกระทบกับส่วนอื่น ๆ ของโปรแกรม

5.4 การประกาศและการเรียกใช้ฟังก์ชันที่สร้างขึ้นมาใช้งานเอง (User defined functions)

หลักกการทำงานของการประกาศและการเรียกใช้ฟังก์ชัน แสดงดังภาพ 5.1



ภาพ 5.1 การประกาศและการเรียกใช้ฟังก์ชัน

ตัวอย่างการประกาศและการเรียกใช้ฟังก์ชัน

โปรแกรม

```
1.  #include <stdio.h>
2.  int addNumbers(int val1,int val2)   // การประกาศชื่อฟังก์ชัน
3.  {
4.      int result;                     //คำสั่งที่ 1 ของฟังก์ชัน
5.      result = val2+val2;             //คำสั่งที่ 2 ของฟังก์ชัน
6.      return result;                  //การส่งค่าจากฟังก์ชันกลับ (return statement)
7.  }
8.  main()
9.  {
10.     int num1,num2,sum;
11.     printf("Enters two numbers: ");
12.     scanf("%d %d",&num1,&num2);
13.     sum = addNumbers(num1, num2); // คำสั่งใน main ที่เรียกฟังก์ชัน (function call)
14.     printf("sum = %d",sum);
15. }
```

5.5 การสร้างฟังก์ชัน

ฟังก์ชันประกอบด้วย 2 ส่วน ได้แก่ ส่วนหัวของฟังก์ชัน (Function Header) และตัวฟังก์ชัน (Body of the Function)

5.5.1 ส่วนหัวของฟังก์ชัน (Function Header)

ส่วนหัวของฟังก์ชันประกอบด้วย 3 ส่วน ได้แก่ 1) ชนิดข้อมูลที่จะส่งกลับ เป็นส่วนที่กำหนดประเภทข้อมูลที่ส่งกลับไปยังส่วนที่มีการเรียกฟังก์ชันหลังจากประมวลผลคำสั่งในฟังก์ชันเสร็จเรียบร้อย ส่วนนี้จะมีหรือไม่มีก็ได้ 2) ชื่อฟังก์ชัน การตั้งชื่อฟังก์ชัน ใช้หลักการเดียวกับการตั้งชื่อตัวแปร และ 3) ส่วนการรับข้อมูลเข้ามาในฟังก์ชัน ประกอบด้วย ชนิดข้อมูล ตามด้วยชื่ออาร์กิวเมนต์ (argument) หรือ ชื่อพารามิเตอร์ (parameter) ที่รับเข้ามาทำงานภายในฟังก์ชัน ส่วนนี้จะมีหรือไม่มีก็ได้เช่นกัน

ชนิดข้อมูลที่จะส่งกลับ ชื่อฟังก์ชัน (ชนิดข้อมูล อาร์กิวเมนต์1,ชนิดข้อมูล อาร์กิวเมนต์2,...){ }

5.5.2 ตัวฟังก์ชัน (Body of the Function)

ชุดคำสั่งที่อยู่ภายในฟังก์ชันอยู่ถัดจากส่วนหัวของฟังก์ชันจะอยู่ภายในบล็อก ที่ปิดหัวท้ายด้วยเครื่องหมาย {...} หากมีการส่งค่ากลับจากฟังก์ชันที่ท้ายฟังก์ชันจะมีคำสั่ง return (ค่าตัวแปรหรือค่าคงที่); สำหรับส่งค่ากลับ

5.5.3 ตัวอย่างการสร้างฟังก์ชัน

ฟังก์ชันชื่อ max ที่ส่งค่ากลับมาเป็นจำนวนเต็ม มีการรับค่าเข้ามาเป็น จำนวนเต็ม 3 ค่าได้แก่ อาร์กิวเมนต์ num1, อาร์กิวเมนต์ num2 และ อาร์กิวเมนต์ num3 และหลังจากประมวลผลเสร็จต้องส่งค่า value ที่มีชนิดข้อมูลเป็นจำนวนเต็มกลับไปยังคำสั่งที่มีการเรียกใช้งานฟังก์ชัน max ดังภาพ 5.2

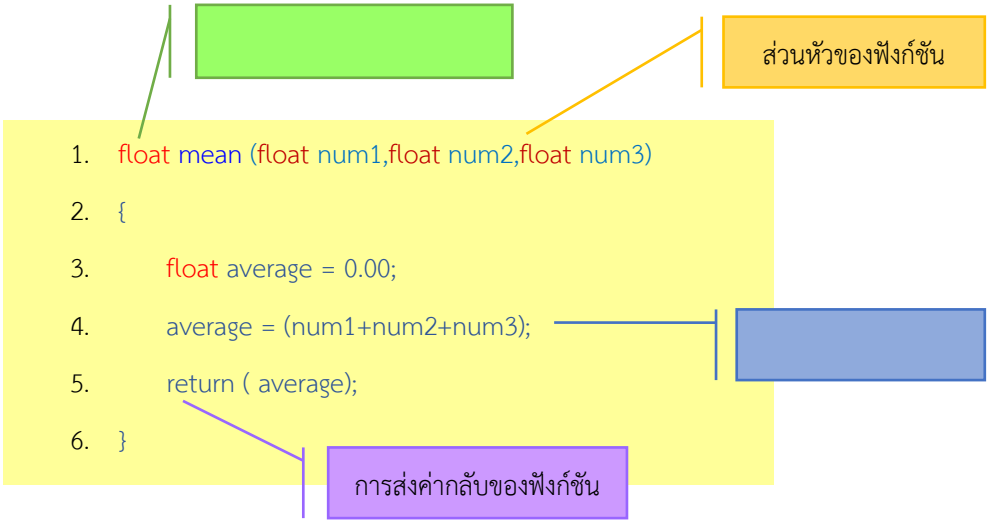
```
int max (int num1, int num2, int num3)
{
        F1
        F2
        .....
        Fn
        return value;
}
```

ภาพ 5.2 ตัวอย่างการประกาศฟังก์ชัน

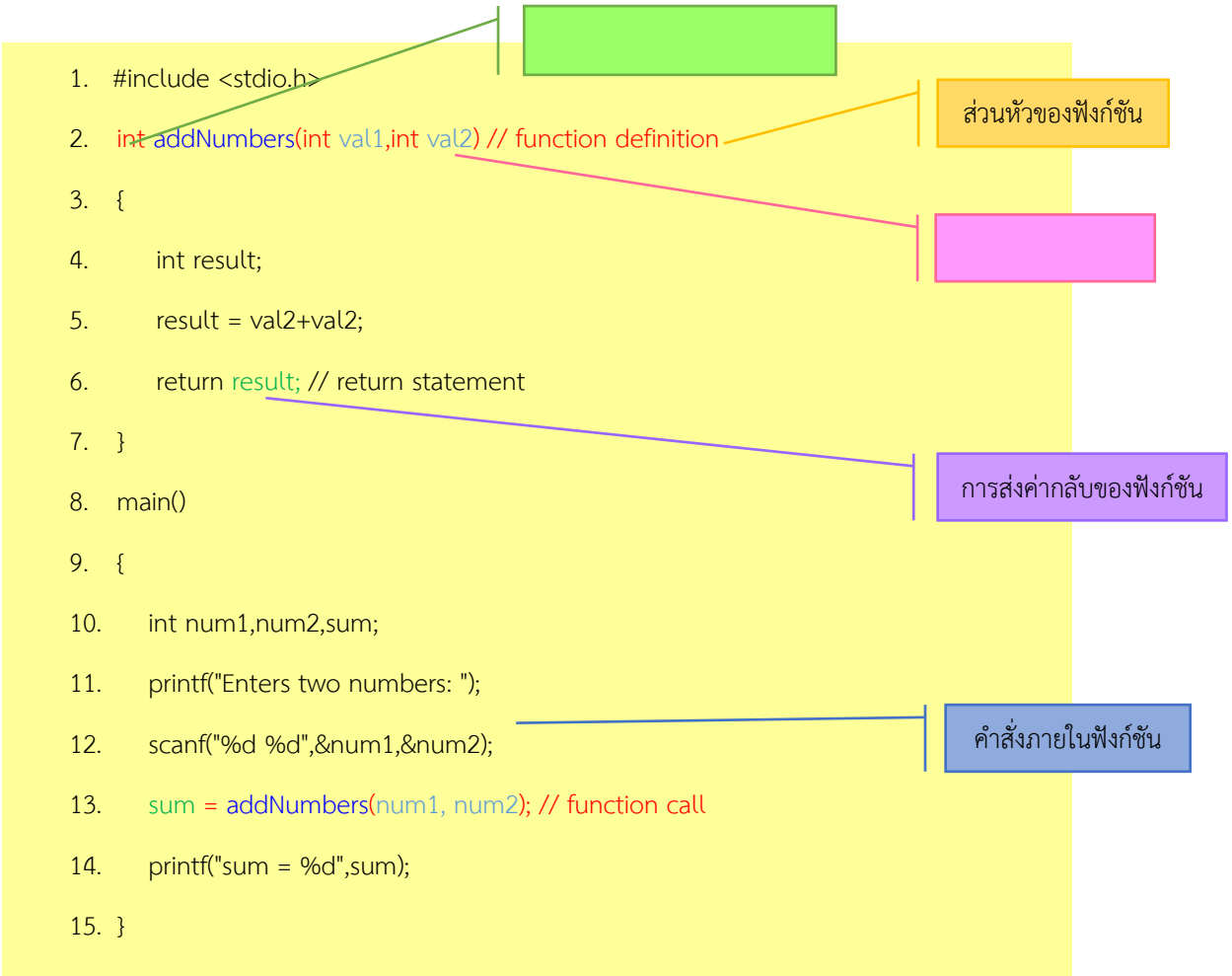
5.6 การส่งผ่านค่าระหว่างฟังก์ชัน

การรับส่งข้อมูลระหว่างฟังก์ชันจะดำเนินการผ่าน อาร์กิวเมนต์ (argument) หรือ พารามิเตอร์ (parameter) เช่น คำสั่งภายใน main() เรียกใช้คำสั่ง mean(val1, val2, val3); เป็นการส่งผ่านค่า val1, val2, val3 เข้าไปในฟังก์ชัน mean() รายละเอียดดังภาพ 5.3 และ 5.4

ตัวอย่างการส่งผ่านค่าระหว่างฟังก์ชัน



ภาพ 5.3 การประกาศและคำสั่งภายในฟังก์ชัน



ภาพ 5.4 ฟังก์ชันและการเรียกใช้งาน

5.7 การประกาศฟังก์ชันต้นแบบ (Function Prototype)

Function Prototype คือ การประกาศรายชื่อฟังก์ชันเพื่อให้คอมไพเลอร์รู้ก่อนว่าภายในโปรแกรมจะมีการเรียกใช้ฟังก์ชันอะไรบ้าง ใช้ในกรณีที่เขียนฟังก์ชันอยู่หลังฟังก์ชัน main( ) จะต้องประกาศ Function Prototype ก่อนฟังก์ชัน main( ) รูปแบบคล้ายกับการประกาศส่วนหัวของฟังก์ชัน คือ มีชื่อฟังก์ชัน และจำนวนอาร์กิวเมนต์ การเขียนฟังก์ชันสามารถเขียนได้ 2 ตำแหน่ง คือ

1) การเขียนฟังก์ชันก่อน main() => สามารถเขียนโปรแกรมส่วนของฟังก์ชันได้ทันที ดังโปรแกรมด้านล่าง

```
1.  #include <stdio.h>
2.  int addNumbers(int val1,int val2)
3.  {
4.      int result;
5.      result = val2+val2;
6.      return result;
7.  }
8.  main()
9.  {
10.     int num1,num2,sum;
11.     printf("Enters two numbers: ");
12.     scanf("%d %d",&num1,&num2);
13.     sum = addNumbers(num1, num2);
14.     printf("sum = %d",sum);
15. }
```

2) การเขียนฟังก์ชันหลัง main() ต้องประกาศ Function Prototype ก่อน main() และเขียนโปรแกรมส่วนของฟังก์ชันหลัง main() ดังโปรแกรมด้านล่าง

```
1.  #include <stdio.h>
2.  int addNumbers(int val1, int val2);
3.  //การประกาศ function prototype
4.  main()
5.  {
6.      int num1,num2,sum;
7.      printf("Enters two numbers: ");
8.      scanf("%d %d",&num1,&num2);
9.      sum = addNumbers(num1, num2);
10.     printf("sum = %d",sum);
11. }
```

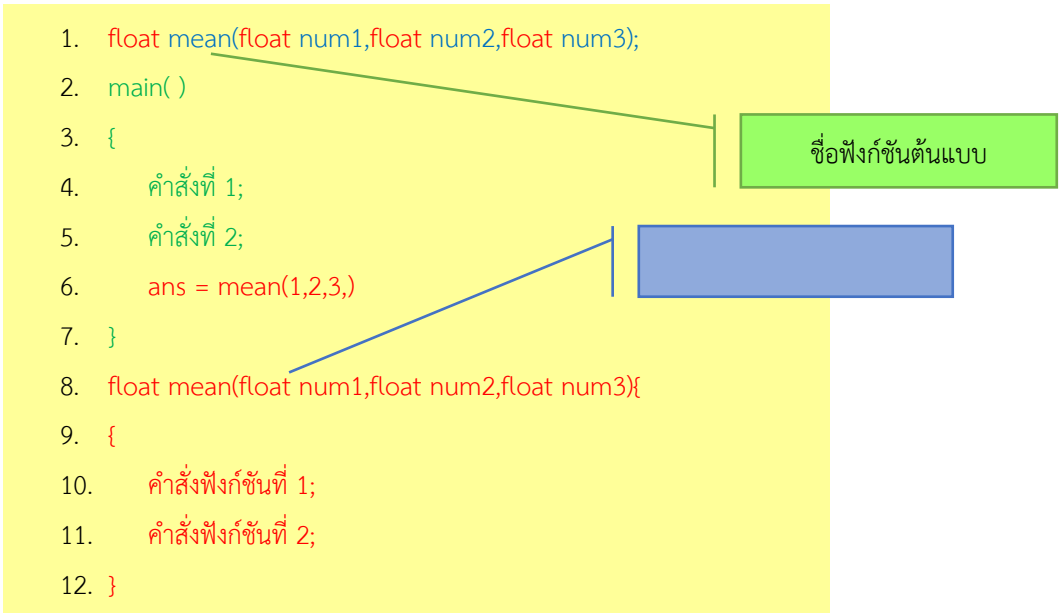
```
12. int addNumbers(int val1,int val2)
13. {
14.     int result;
15.     result = val2+val2;
16.     return result
17. }
```

5.7.1 รูปแบบการประกาศฟังก์ชันต้นแบบ (Function Prototype)

1) การประกาศแบบเต็ม มีรูปแบบการประกาศดังต่อไปนี้ โดยตัวอย่างแสดงดังภาพ 5.4

รูปแบบการประกาศแบบเต็ม
ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน (ชนิดข้อมูล อาร์กิวเมนต์1,ชนิดข้อมูล อาร์กิวเมนต์2,...);
ตัวอย่าง
int max(int num1, int num2, int num3);

ตัวอย่างการประกาศฟังก์ชันต้นแบบแบบเต็ม

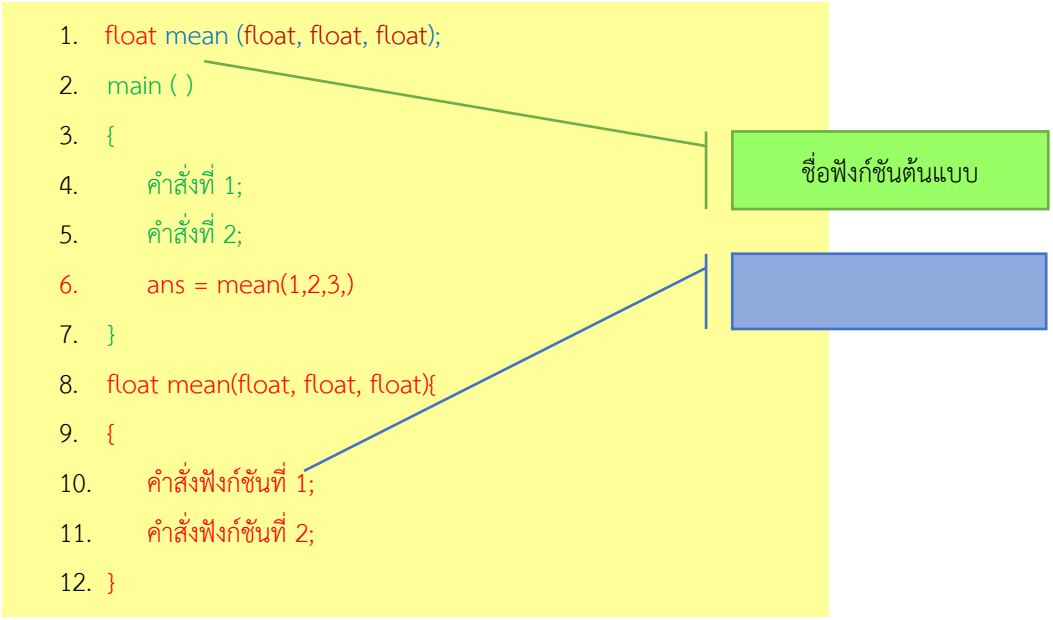


ภาพ 5.5 ตัวอย่างการประกาศฟังก์ชันต้นแบบ (แบบเต็ม)

2) การประกาศแบบย่อ มีรูปแบบการประกาศดังต่อไปนี้ โดยตัวอย่างแสดงดังภาพ 5.5

รูปแบบการประกาศแบบย่อ
ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน (ชนิดข้อมูล1 ,ชนิดข้อมูล2 ,...);
ตัวอย่าง
int max(int, int, int);

ตัวอย่างการประกาศฟังก์ชันต้นแบบแบบย่อ



ภาพ 5.6 ตัวอย่างการประกาศฟังก์ชันต้นแบบ (แบบย่อ)

5.8 การเข้าถึงฟังก์ชัน (Accessing a Function)

การเรียกใช้งานฟังก์ชันทำได้โดยเรียกผ่านชื่อฟังก์ชัน ตามด้วยอาร์กิวเมนต์ที่ต้องการส่งผ่านไปยังฟังก์ชันถ้ามี หรือถ้าไม่มีอาร์กิวเมนต์ก็ใส่วงเล็บเปล่า ๆ “( )” เช่น mean(3,5,7); test( ); ชนิดของการเข้าถึงฟังก์ชันแบ่งเป็น 4 ชนิด ได้แก่

5.8.1 ฟังก์ชันที่ไม่มีการส่งผ่านค่าใด ๆ (Function with no arguments and no return value)

จัดเป็นฟังก์ชันธรรมดา ไม่มีการส่งผ่านอาร์กิวเมนต์ใด ๆ ฟังก์ชันจะมุ่งเน้นทำงานอย่างใดอย่างหนึ่งให้เสร็จสิ้นเท่านั้นไม่มีการส่งกลับค่าใด ๆ ออกจากฟังก์ชัน ซึ่งคำสั่ง void จะใส่หรือไม่ใส่ก็ได้ในการประกาศฟังก์ชัน

รูปแบบฟังก์ชันที่ไม่มีการส่งผ่านค่า	ตัวอย่าง

แบบที่ 1
void ชื่อฟังก์ชัน (void)
{
statement-f1;
statement-f2;
.....;
statement-fn;
}
	void line (void)
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	}

แบบที่ 2
void ชื่อฟังก์ชัน ()
{
statement-f1;
statement-f2;
.....;
statement-fn;
}
	void line ()
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	}

แบบที่ 3
ชื่อฟังก์ชัน ()
{
statement-f1;
statement-f2;
.....;
statement-fn;
}
	line ()
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	}

ตัวอย่างการเรียกฟังก์ชันที่ไม่มีการส่งผ่านค่า

โปรแกรม	ผลลัพธ์

1. #include <stdio.h>
2. void line ( )
3. {
4. int i;
5. for (i=0 ;i<= 5;i++)
6. printf (":\");
7. }
8. main ( )
9. {
10. line ( );
11. }	

5.8.2 ฟังก์ชันที่ส่งผ่านค่าทางเดียว แต่ไม่มีการส่งค่ากลับ (Function with arguments and no return value)

เป็นฟังก์ชันที่มีการรับอาร์กิวเมนต์เข้ามาตอนถูกเรียกใช้งานฟังก์ชันจะนำอาร์กิวเมนต์ที่รับมาไปใช้ในการทำงานอย่างใดอย่างหนึ่งให้เสร็จสิ้นเท่านั้นไม่มีการส่งกลับค่าใด ๆ ออกมาจากฟังก์ชัน

รูปแบบฟังก์ชันที่ส่งผ่านค่าทางเดียว แต่ไม่มีการส่งค่ากลับ	**ตัวอย่าง**

void ชื่อฟังก์ชัน (ชนิดข้อมูล อาร์กิวเมนต์1, ชนิดข้อมูล อาร์กิวเมนต์2,...)
{
statement-f1;
statement-f2;
.....;
statement-fn;
}
	void bar (int amount)
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	}

ตัวอย่างการเรียกฟังก์ชันที่ส่งผ่านค่าทางเดียว แต่มีการส่งค่ากลับ

โปรแกรม	**ผลลัพธ์**

1. #include <stdio.h>
2. void bar (int amount)
3. {
4. int i,count =1;
5. do {
6. for (i=0 ;i<= amount ;i++)
7. printf ("%c",65);
8. printf ("\n");
9. count++;
10. } while (count <=3);
11. printf ("%d\n\n",amount);
12. }
13. main ( )
14. {
15. int num = 5;
16. bar (num);
17. }	

5.8.3 ฟังก์ชันที่ไม่มีการรับค่า แต่ส่งค่ากลับออกไป (Function with no arguments and a return value)

เป็นฟังก์ชันย่อยที่ไม่ได้รับค่าใด ๆ จากฟังก์ชันที่เรียกใช้ แต่เมื่อมีการประมวลผลใด ๆ ภายในฟังก์ชันเสร็จแล้ว จะมีการส่งผลลัพธ์จากการทำงานกลับไปยังฟังก์ชันหลักที่เรียกใช้ฟังก์ชันย่อยนี้ การส่งค่ากลับจะทำได้โดยการใช้คำสั่ง return ตามด้วยค่าที่ต้องการส่งกลับไป โดยค่าที่จะ return ไปสามารถเป็นได้ทั้งนิพจน์ (expression), ค่าตัวแปร (variable) และค่าคงที่ (value) ดังตัวอย่างด้านล่าง

<u>รูปแบบการ return</u>

return (expression); เช่น return(num1+num2);

return (variable); เช่น return(number);

return (value); เช่น return(10);

ฟังก์ชันที่ไม่มีการรับค่า แต่ส่งค่ากลับออกไป	ตัวอย่าง

ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน (void)
{
statement-f1;
statement-f2;
.....;
return ค่าที่ส่งกลับออกไป;
}
	float getVat (void)
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	return (value);
	}

ตัวอย่างฟังก์ชันที่ไม่มีการรับค่า แต่ส่งค่ากลับออกไป

โปรแกรม

```
1.  #include <stdio.h>
2.  float getVat (void);
3.  main () {
4.      float vat, price = 1000;
5.      vat = (getVat() * price) / 100;
6.      printf ("Price (%.2f baht) include VAT is %.2f baht", price, price + vat);
7.  }
8.  float getVat (void) {
9.      float vatTH = 7;
10.     return vatTH;
11. }
```

ผลลัพธ์

5.8.4 ฟังก์ชันที่ส่งผ่านค่าทั้งไปและกลับ (Function with arguments and a return value)

เป็นฟังก์ชันย่อยที่มีการรับค่าจากฟังก์ชันที่เรียกใช้ และเมื่อมีการประมวลผลใด ๆ ภายในฟังก์ชันเสร็จแล้ว จะมีการส่งผลลัพธ์จากการทำงานกลับไปยังฟังก์ชันหลักที่เรียกใช้ฟังก์ชันนี้ การส่งค่ากลับจะทำได้โดยการใช้คำสั่ง return ตามด้วยค่าที่ต้องการส่งกลับไป

ฟังก์ชันที่ส่งผ่านค่าทั้งไปและกลับ	ตัวอย่าง

ชนิดข้อมูลที่ส่งกลับ ชื่อฟังก์ชัน (ชนิดข้อมูล อาร์กิวเมนต์1, ชนิดข้อมูล อาร์กิวเมนต์2,...)
{
statement-f1;
statement-f2;
.....;
return ค่าที่ส่งกลับออกไป;
}
	float getVat (float vatJP)
	{
	statement-f1;
	statement-f2;
	.....;
	statement-fn;
	return (value);
	}

ตัวอย่างฟังก์ชันที่ส่งผ่านค่าทั้งไปและกลับ

โปรแกรม

```
1.  #include <stdio.h>
2.  float getVat (float vatJP);
3.  main () {
4.      float vat = 7, total_price, price = 1000;
5.      total_price = (getVat(10.00) * price) / 100;
6.      printf ("Price (%.2f baht) include VAT is %.2f baht", price, total_price);
7.  }
8.  float getVat (float vatJP) {
9.      return vatJP;
10. }
```

ผลลัพธ์

5.9 ชนิดของตัวแปรโดยแบ่งตามวิธีการจัดเก็บของคอมไพล์เลอร์ (storage class)

แบ่งเป็น 4 ชนิด ได้แก่ Automatic, External, Static และ Register variable

5.9.1 Automatic variable

Automatic variable หรือเรียกว่า Local variable เป็นตัวแปรชนิดค่าเริ่มต้น (default) เมื่อมีการประกาศตัวแปรภายในฟังก์ชันแล้ว ค่าของตัวแปรชนิดนี้จะคงอยู่ เมื่อการเรียกใช้ฟังก์ชันที่ประกาศค่าตัวแปรนั้นไว้เท่านั้น และค่าของตัวแปรภายในฟังก์ชันนี้จะหายไปเมื่อสิ้นสุดการทำงานของฟังก์ชันแล้ว การประกาศตัวแปรแบบ Automatic variable นี้ให้ใช้ keyword ว่า auto นำหน้าชนิดข้อมูล

รูปแบบการประกาศ Automatic variable	ตัวอย่าง
auto ชนิดข้อมูล ชื่อตัวแปร;	auto int num;

ตัวอย่างการใช้ Automatic variable แบบที่ 1

โปรแกรม	ผลลัพธ์

1. #include <stdio.h>
2. int sum(int num1, int num2){
3. auto int result; //ประกาศ Automatic variable
4. result= num1+num2;
5. return result;
6. }
7. main()
8. {
9. int val1 = 2, val2 = 3, result= 10,add;
10. add = sum(val1,val2);
11. printf("sum is : %d\n", add);
12. printf("result is : %d\n", result);
13. }	

ตัวอย่างการใช้ Automatic variable แบบที่ 2

โปรแกรม	ผลลัพธ์

1. #include<stdio.h>
2. display();
3. main ()
4. {
5. auto int num=10; //main int num=10;
6. printf("Num : %d\n", num);
7. display ();
8. printf("Num : %d\n", num);
9. }
10. display()
11. {
12. int num=20; //function auto int num=20;
13. printf("Num : %d\n", num);
14. }	

5.9.2 External variable

External variable หรือเรียกว่า Global variable คือ ตัวแปรที่ประกาศไว้แล้วทุกฟังชัน รวมถึง main() สามารถเรียกใช้งานได้ด้วยตัวแปรแบบ extern จะคงอยู่ตลอดการทำงานของโปรแกรมและทุก function จะสามารถมองเห็นและเรียกใช้ได้ การประกาศตัวแปรแบบ External variable นี้ให้ใช้ keyword ว่า extern หรือไม่ระบุ keyword ก็ได้ นำหน้าชนิดข้อมูล

รูปแบบการประกาศ External variable	ตัวอย่าง
extern ชนิดข้อมูล ชื่อตัวแปร; หรือ ชนิดข้อมูล ชื่อตัวแปร;	extern int num; หรือ int num;

ตัวอย่างการใช้ External variable แบบที่ 1

โปรแกรม	ผลลัพธ์

1. #include <stdio.h>
2. extern int num =5;
3. display() {
4. int i;
5. for(i=0;i<=num;i++)
6. printf("num = %d\n", num);
7. }
8. main()
9. {
10. num = 3;
11. printf("---num = %d---\n", num);
12. display();
13. }	

ตัวอย่างการใช้ External variable แบบที่ 2

โปรแกรม	ผลลัพธ์

1. #include<stdio.h>
2. increment();
3. decrement();
4. display();
5. int num=10; //global variable
6. main()
7. {
8. printf("Num : %d\n",num);
9. increment(); display();
10. decrement(); display();
11. printf("Num : %d\n",num);
12. }
13. increment() { num = 20; }
14. decrement() { num = 5; }
15. display() { printf("Num : %d\n",num); }	

5.9.3 Static variable

ตัวแปรแบบ Static จะมีขอบเขตการเรียกใช้เหมือนกับตัวแปรแบบ automatic หรือ local variable ค่าของตัวแปรแบบ static นี้ จะไม่หายไปเมื่อจบ function นั้น ๆ เมื่อมีการเรียกค่าของตัวแปรนี้จาก function ที่เคยเรียกอีกครั้ง ค่าของตัวแปรจะยังคงอยู่และทีค่าเท่ากับการเรียกใช้ตัวแปรครั้งก่อน การประกาศตัวแปรแบบ Static variable นี้ให้ใช้ keyword ว่า static นำหน้าชนิดข้อมูล

รูปแบบการประกาศ Static variable	ตัวอย่าง
static ชนิดข้อมูล ชื่อตัวแปร;	static int num;

ตัวอย่างการใช้ Static variable แบบที่ 1

โปรแกรม	ผลลัพธ์

1. #include <stdio.h>
2. display();
3. main()
4. {
5. display();
6. display();
7. }
8. display ()
9. {
10. static int data = 0;
11. printf("data = %d \n",data);
12. data += 5;
13. }	

ตัวอย่างการใช้ Static variable แบบที่ 2

โปรแกรม	ผลลัพธ์

1. #include <stdio.h>
2. static int num = 5;
3. display(){
4. static int i = 0;
5. printf ("num = %d\t", num--);
6. printf ("i = %d\n",i++);
7. }
8. main (){
9. while (num >= 2)
10. display ();
11. }	

5.9.4 Register variable

ปกติเมื่อมีการประกาศตัวแปรแล้ว ตัวแปรนั้นจะถูกเก็บในหน่วยความจำ แต่ถ้าประกาศตัวแปรให้เป็นแบบ register ตัวแปรนั้นจะถูกเก็บลงใน register ของ CPU ทำให้ทำงานได้เร็วขึ้น ปัจจุบันการประกาศตัวแปรชนิด register variable ไม่ได้เร็วกว่าตัวแปรปกติที่ถูกเก็บในหน่วยความจำมากนัก เนื่องจากคอมพิวเตอร์มีสิทธิภาพมากขึ้น คุณสมบัติของตัวแปรชนิด register variable จะเหมือนกับตัวแปร ชนิด automatic variable คือ ตัวแปรจะสามารถใช้งานได้เฉพาะฟังก์ชันที่ประกาศตัวแปรไว้ หลังจากทำงานเสร็จค่าของตัวแปรจะถือว่าสิ้นสุด การประกาศตัวแปรแบบ Static variable นี้ให้ใช้ keyword ว่า register นำหน้าชนิดข้อมูล

ตัวอย่างการใช้ Register variable แบบที่ 1

โปรแกรม	ผลลัพธ์
1. #include<stdio.h> 2. main() 3. { 4. register int data=10; 5. printf("Data : %d",data); 6. }	

คำถามท้ายบทที่ 5

1. จงอธิบายความหมายของฟังก์ชันมาตรฐานในภาษา C ดังนี้
   1.1. ฟังก์ชัน strcpy
   1.2. ฟังก์ชัน sqrt
   1.3. ฟังก์ชัน strlen
   1.4. ฟังก์ชัน pow
   1.5. ฟังก์ชัน floor
2. จงอธิบายความหมายของไลบรารี่ ctype.h, string.h, math.h และยกตัวอย่างฟังก์ชันของแต่ละไลบรารี่มาอย่างละ 3 ฟังก์ชัน
3. จงอธิบายความหมายของฟังก์ชันต้นแบบ
4. จงเขียนโปรแกรมรับข้อความ 2 ครั้ง แล้วใช้ฟังก์ชันมาตรฐาน tolower() และ toupper แล้วแสดงผลลัพธ์โดยให้อยู่ในรูปแบบดังนี้

```
Input String : KASETSART UNIVERSITY
Lower String : kasetsart university
Input String : sriracha campus
Upper String : SRIRACHA CAMPUS
```

5. จงเขียนโปรแกรมแปลงอุณหภูมิในหน่วยองศาเซลเซียสให้เป็นองศาฟาเรนต์ไฮท์ แล้วแสดงผลลัพธ์โดยให้อยู่ในรูปแบบดังนี้

```
Celsius  : _____
Fahrenheit : _____
```

6. จงเขียนโปรแกรมโดยสร้างฟังก์ชันคำนวณพื้นที่วงกลม รับค่ารัศมีของวงกลม แล้วแสดงผลลัพธ์โดยให้อยู่ในรูปแบบดังนี้

```
Input Radius : _____
Circle Area : _____
```

7. จงเขียนโปรแกรมเพื่อรับค่าจำนวนเต็มจำนวน 5 ค่า และแสดงค่าที่น้อยที่สุดและมากที่สุด โดยกำหนดให้ในโปรแกรมมีฟังก์ชันอย่างน้อย 2 ฟังก์ชัน
8. จงเขียนโปรแกรมเพื่อรับค่าจำนวนเต็มจำนวน 5 ค่า และแสดงค่าที่น้อยที่สุดและมากที่สุด โดยกำหนดให้ในโปรแกรมมีฟังก์ชันอย่างน้อย 2 ฟังก์ชันและต้องมีการประกาศ Function Prototype ด้วย
9. จงเขียนโปรแกรมเพื่อรับค่าจำนวนเต็มจำนวน 1 ค่า และตรวจสอบค่าดังกล่าวว่ามีคุณสมบัติเป็นค่า Prime number หรือไม่ ถ้ามีคุณสมบัติเป็น Prime number ให้แสดงว่า “X is Prime Number” แต่ถ้าไม่มีคุณสมบัติเป็น Prime number ให้แสดงว่า “X is not Prime Number” โดยกำหนดให้เขียนฟังก์ชันแบบฟังก์ชันที่รับค่า และส่งค่ากลับออกไป (Function with arguments and a return value) เพื่อตรวจสอบค่า Prime number ดังกล่าว
10. จงอธิบายความแตกต่างของ Automatic Variable, External Variable, Static Variable และ Register Variable