บทที่ 11 ฟังก์ชัน I

Function I

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ฟังก์ชัน

1. ฟังก์ชันคืออะไร
2. ประเภทของฟังก์ชัน
3. รูปแบบโครงสร้างของฟังก์ชัน
4. การรับค่าและส่งค่าจากฟังก์ชัน
5. ตัวแปรและขอบเขตการใช้งานของฟังก์ชัน
6. การส่งค่าตัวแปร (pass by value & pass by reference)

ฟังก์ชันคืออะไร

- ฟังก์ชัน คือ ชุดของการทำงาน ที่ถูกเขียนขึ้นให้โปรแกรมเมอร์สามารถเรียกใช้งานได้โดยง่าย

** ฟังก์ชัน คือ ชุดของการทำงาน

** ฟังก์ชัน ถูกเรียกใช้งานได้

การพัฒนาโปรแกรมกับฟังก์ชัน

ปัญหาที่เกิดขึ้นในการพัฒนาโปรแกรมที่ซับซ้อน

- โปรแกรมเมอร์ไม่สามารถทราบการทำงานของระบบโดยละเอียดได้เช่น ไม่ทราบกระบวนการส่งข้อมูลผ่านเครือข่าย แต่ต้องเขียนโปรแกรมเพื่อเชื่อมต่อระบบเครือข่าย
- โปรแกรมเมอร์ไม่สามารถทราบขั้นตอนการทำงานของคอมพิวเตอร์ทั้งหมดได้ เช่น ทำอย่างไรตัวอักษรจึงปรากฏในหน้าจอภาพได้
- โปรแกรมบางโปรแกรมมีการทำงานที่ซับซ้อน และการทำงานซับซ้อนนั้นถูกเรียกใช้งานบ่อยครั้งเช่น การหาผลลัพธ์ทางวิทยาศาสตร์ การวิเคราะห์ข้อมูลขนาดใหญ่เป็นต้น

การพัฒนาโปรแกรมกับฟังก์ชัน

วิธีการแก้ไข

- เพื่อให้โปรแกรมเมอร์สามารถทำงานได้โดยไม่จำเป็นต้องทราบรายละเอียดการทำงานทั้งหมดของระบบ จะให้โปรแกรมเมอร์ที่ทราบการทำงานโดยละเอียดของกระบวนการต่างๆ จะเขียนชุดคำสั่งในรูปแบบของ **ฟังก์ชัน** แล้วแจกจ่ายให้โปรแกรมเมอร์อื่นๆ ได้ใช้งาน
- โปรแกรมเมอร์สามารถเรียกใช้ฟังก์ชันโดยทราบเพียงวิธีการใช้งาน และผลลัพธ์ที่เกิดขึ้นหลังจากเรียกใช้งานฟังก์ชันเท่านั้น
  - เช่น โปรแกรมเมอร์ที่ไม่ทราบว่าทำอย่างไรตัวอักษรจึงจะปรากฏหน้าจอ สามารถใช้คำสั่ง printf ได้เลย โดยโปรแกรมเมอร์จะทราบเพียงแค่ การเรียก printf จะทำให้มีตัวอักษรปรากฏบนหน้าจอได้เท่านั้น

ข้อดีของฟังก์ชัน

- ทำให้โปรแกรมเมอร์สามารถพัฒนาโปรแกรมได้โดยง่าย โดยโปรแกรมเมอร์ไม่จำเป็นต้องทราบว่าการทำงานของฟังก์ชันทำงานอย่างไรทราบเพียงผลลัพธ์ของการทำงานเท่านั้น
- โปรแกรมเมอร์สามารถเขียนโปรแกรมให้มีการทำงานที่ซับซ้อนได้ โดยไม่จำเป็นต้องเขียนโปรแกรมส่วนที่ซับซ้อนนั้นหลายๆ ครั้ง
- โปรแกรมเมอร์สามารถออกแบบฟังก์ชันได้ตามความต้องการของโปรแกรมเมอร์

ฟังก์ชัน

1. ฟังก์ชันคืออะไร
2. ประเภทของฟังก์ชัน
3. รูปแบบโครงสร้างของฟังก์ชัน
4. การรับค่าและส่งค่าจากฟังก์ชัน
5. ตัวแปรและขอบเขตการใช้งานของฟังก์ชัน
6. การส่งค่าตัวแปร (pass by value & pass by reference)

ฟังก์ชัน (Function)แบ่งออกเป็น 2 ประเภท

1. ฟังก์ชันไลบรารีมาตรฐาน (Standard Library function)
2. ฟังก์ชันที่สร้างขึ้นเอง (User Defined function)

ฟังก์ชันไลบรารีมาตรฐาน (Standard Library Function)

- ฟังก์ชันไลบรารีมาตรฐาน (Standard Library Function) เป็นฟังก์ชันที่มีอยู่แล้วเก็บไว้ใน Library ในการใช้งานต้อง include directives ก่อน
- directive คือสารบัญของกลุ่มฟังก์ชัน เช่น stdio.h , conio.h , string.h , math.h เป็นต้น
- การ include directives จะเป็นเหมือนการประกาศให้กับ compiler ทราบว่าจะใช้คำสั่ง ในกลุ่มของ directive นั้นๆ เช่น การใช้คำสั่ง sin() ซึ่งอยู่ใน math.h จะต้องมีบรรทัด include math.h เสมอ ดังตัวอย่าง

ตัวอย่าง


```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
   double rad = -1.0;
   do {
       printf ( "Sine of %f is %f\n", rad, sin(rad));
       rad += 0.1;
   } while (rad <= 1.0);
   return 0;
}
```

การเรียกใช้ Standard Library Function

- การเรียกใช้ Standard Library Function มีขั้นตอนดังนี้
  - ทราบว่าโปรแกรมที่เขียนต้องการการทำงานอะไร
  - การทำงานดังกล่าวคือฟังก์ชันชื่ออะไร
  - ทราบ directive ที่เป็นสารบัญของคำสั่ง
  - Include directive นั้นๆ
  - เรียกใช้ฟังก์ชันในโปรแกรม

ตัวอย่างไลบรารีฟังก์ชัน

- **ฟังก์ชันการคำนวณทางคณิตศาสตร์**
  - ไฟล์ header => math.h
- **ฟังก์ชันสำหรับอักขระและข้อความ**
  - ไฟล์ header => string.h

    ctype.h

ฟังก์ชันการคำนวณทางคณิตศาสตร์


```
#include<math.h>
```

```
sin(var);
cos(var);
tan(var);
sqrt(var);//รากที่ 2
pow(var1,var2);//var1 ยกกำลัง var2
log(var);// log ฐาน e
log10(var);// log ฐาน 10
exp(var);// e ยกกำลัง
fabs(var);// ค่าสัมบูรณ์แบบfloat
```

ตัวอย่างการใช้งานฟังก์ชันการคำนวณทางคณิตศาสตร์


```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define PI 3.14
int main()
{
   float deg,rad;
   printf ("Enter Degree : ");
   scanf ("%f",&deg);
   rad = deg * PI / 180;
   printf ("sin(%.2f) = %.3f\n",deg,sin(rad));
   printf ("cos(%.2f) = %.3f\n",deg,cos(rad));
   printf ("tan(%.2f) = %.3f\n",deg,tan(rad));
   return 0;
}
```

ฟังก์ชันสำหรับข้อความ


```
#include<string.h>
```

```
strcpy(str1, str2);
strcat(dest1, src2);
strcmp(dest1, src2);
strcmpi(str1, str2);
strlen(str);
```

```
#include<ctype.h>
```

```
tolower(ch);
toupper(ch);
```

ฟังก์ชันที่สร้างขึ้นเอง (User-defined Function)

- เนื่องจาก Standard Library Function ทั้งหมด เป็นฟังก์ชันมาตรฐานที่มีเฉพาะการทำงานพื้นฐานต่างๆ เท่านั้น
- หากต้องการฟังก์ชันที่มีการทำงานเฉพาะกิจ โปรแกรมเมอร์ต้องเขียนฟังก์ชันขึ้นมาเอง

ข้อกำหนดพื้นฐานของ User-defined Function

1. มีการประกาศ function prototype ที่ต้นโปรแกรมเสมอ จึงจะเรียกใช้งาน function นั้นๆ ได้ (เป็นการบอก Compiler ว่าคำสั่งดังกล่าวคือฟังก์ชัน ไม่ใช่ syntax error)
2. ต้องมีการเขียนฟังก์ชันตามโครงสร้างที่ได้ประกาศไว้ใน function prototype เท่านั้น

โครงสร้างโปรแกรมภาษาซี


```
#include<file.h>
type function_name(type);
type variable
int main()
{
    type variable;
    statement-1;
    ...
    statement-n;
    return 0;
}
type function_name(type variable)
{
    statement-1;
    ...
    statement-n;
    return(var);
}
```

พรีโปรเซสเซอร์ไดเร็คทีฟ
ฟังก์ชันโพรโทรไทพ์
ตัวแปรชนิดโกบอล

ส่วนหัวโปรแกรม

ตัวแปรชนิดโลคอล

คำสั่ง

ฟังก์ชันหลัก

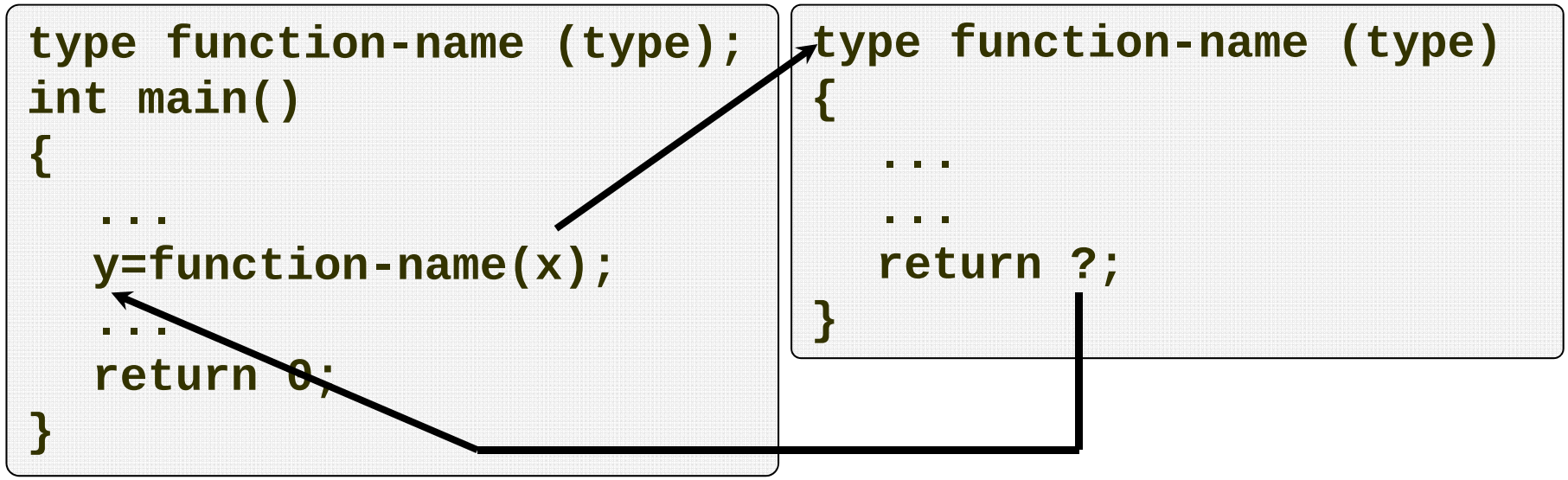
ฟังก์ชันย่อย

ส่วนตัวโปรแกรม

ฟังก์ชัน

1. ฟังก์ชันคืออะไร
2. ประเภทของฟังก์ชัน
3. รูปแบบโครงสร้างของฟังก์ชัน
4. การรับค่าและส่งค่าจากฟังก์ชัน
5. ตัวแปรและขอบเขตการใช้งานของฟังก์ชัน
6. การส่งค่าตัวแปร (pass by value & pass by reference)

ฟังก์ชันที่สร้างขึ้นเอง (User-defined Function)



โปรโตไทป์ฟังก์ชัน (Function Prototype)

เป็นการประกาศการใช้งานฟังก์ชั่นที่อยู่หลัง main()

```
type function_name(type-1,type-2,...,type-n);
```

type	คือ ชนิดของฟังก์ชัน ว่าฟังชันที่ทำการสร้างจะส่งข้อมูลชนิดใดกลับ
function_name	คือ ชื่อฟังก์ชันที่จะสร้างขึ้น
type-n	คือ ชนิดของข้อมูลที่จะส่งให้ฟังก์ชัน

ฟังก์ชันที่สร้างขึ้นเอง (User-defined Function)

การเขียนโปรแกรมโดยมีฟังก์ชันที่สร้างขึ้นเองมี 2 รูปแบบ

- สร้างฟังก์ชัน **ก่อน** ฟังก์ชันหลัก

  ฟังก์ชันหลักสามารถเรียกใช้งานฟังก์ชันที่สร้างขึ้นได้

- สร้างฟังก์ชัน **หลัง** ฟังชันหลัก

  ต้องประกาศ Function Prototype ก่อนเพื่อให้ฟังก์ชันหลักรู้ว่ามีฟังก์ชันที่สร้างขึ้น

สร้างฟังก์ชันก่อนฟังก์ชันหลัก

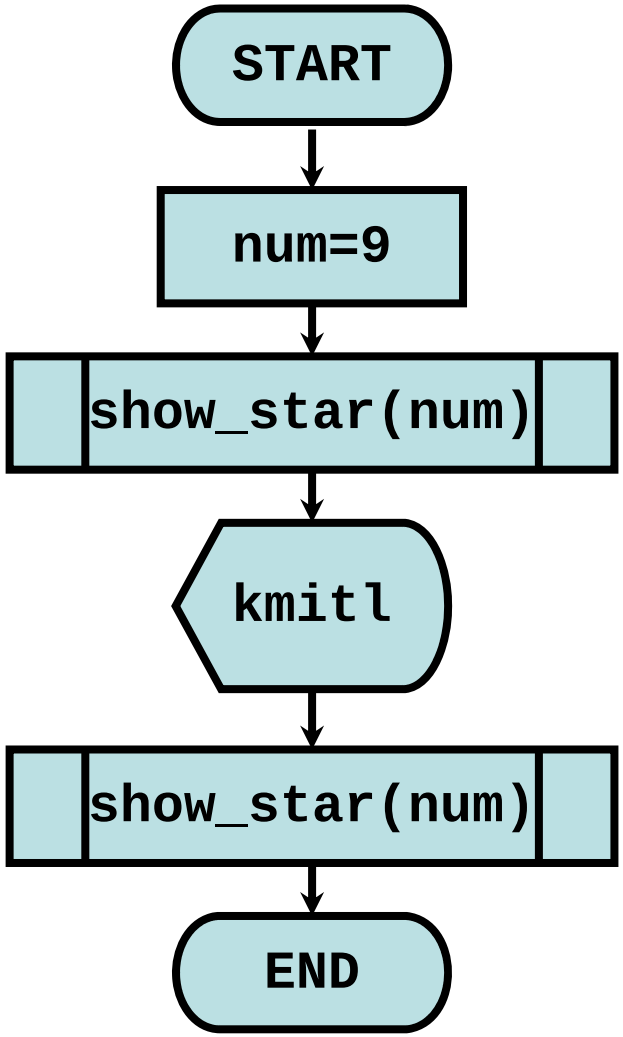

```
#include<file.h>
type variable
type function_name(type variable)
{
    statement-1;
    ...
    statement-n;
    return(var);
}
int main()
{
    type variable;
    statement-1;
    ...
    statement-n;
    return 0;
}
```

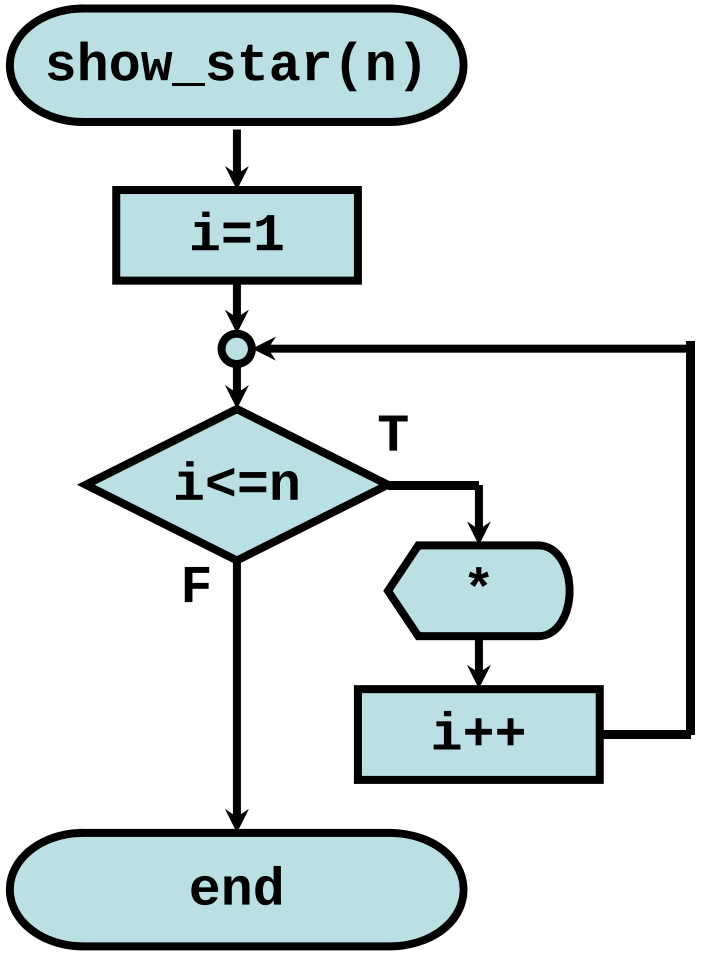
สร้างฟังก์ชันหลังฟังก์ชันหลัก


```
#include<file.h>
type function_name(type);
type variable
int main()
{
    type variable;
    statement-1;
    ...
    statement-n;
    return 0;
}
type function_name(type variable)
{
    statement-1;
    ...
    statement-n;
    return(var);
}
```

ตัวอย่าง | ผังงาน & ผลลัพธ์


```
*********
* kmitl *
*********
```





ตัวอย่าง | โปรแกรม


```
#include<stdio.h>
#include<conio.h>
void show_star (int n)
{
  int i;
  for (i=1;i<=n;i++)
  putchar('*');
}
int main()
{
  int num=9;
  show_star(num);
  printf ("\n* kmitl *\n");
  show_star(num);
  return 0;
}
```

```
#include<stdio.h>
#include<conio.h>
void show_star (int);
int main()
{
  int num=9;
  show_star(num);
  printf ("\n* kmitl *\n");
  show_star(num);
  return 0;
}
void show_star (int n)
{
  int i;
  for (i=1;i<=n;i++)
  putchar('*');
}
```

ฟังก์ชัน

1. ฟังก์ชันคืออะไร
2. ประเภทของฟังก์ชัน
3. รูปแบบโครงสร้างของฟังก์ชัน
4. การรับค่าและส่งค่าจากฟังก์ชัน
5. ตัวแปรและขอบเขตการใช้งานของฟังก์ชัน
6. การส่งค่าตัวแปร (pass by value & pass by reference)

ประเภทของฟังก์ชันที่สร้างขึ้นเอง

- ฟังก์ชันที่ไม่มีการรับส่งค่า
- ฟังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ
- ฟังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ
- ฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ

** สามารถแยกประเภทได้โดยดูจาก function prototype

ฟังก์ชันที่ไม่มีการรับส่งค่า

- เป็นฟังก์ชันที่ไม่มีการรับค่าเข้ามาในฟังก์ชัน และไม่มีการส่งค่ากลับออกไปจากฟังก์ชัน

```
#include<stdio.h>
#include<conio.h>
void function_name(void);
int main()
{
   ...
   function_name();
   ...
   return 0;
}
```

```
void function_name()
{
   statement-1;
   statement-2;
   ...
   statement-n;
}
```

ตัวอย่าง **ฟังก์ชันที่ไม่มีการรับส่งค่า**


```
#include<stdio.h>
void    PrintHello(void);
int main()
{
        PrintHello();
        printf("Hello,in function main.\n");
        PrintHello();
        return 0;
}
void    PrintHello(void)
{
    printf("Hello,in function PrintHello ..\n\n");
}
```

ฟังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ

- เป็นฟังก์ชันที่มีการรับค่าเข้ามาในฟังก์ชัน แต่ไม่มีการส่งค่ากลับออกไปจากฟังก์ชัน

```
#include<stdio.h>
#include<conio.h>
void func_name(type);
int main()
{
   ...
   func_name(varX);
   ...
   return 0;
}
```

```
void func_name(type varY)
{
   statement-1;
   statement-2;
   ...
   statement-n;
}
```

ตัวอย่างฟังก์ชันที่มีการรับค่า แต่ไม่ส่งค่ากลับ

```
#include<stdio.h>
void PrintHello( int );
int main()
{       int n;
        printf("input number of hello : ");
        scanf("%d",&n);
        PrintHello( n );
        return 0;
}
void PrintHello( int i )
{  int count;
   for ( count=1; count<=i ; count++ )
        printf("%d HELLO\n",count);
}
```

ฟังก์ชันที่มีการรับค่า และมีการส่งค่ากลับ

- เป็นฟังก์ชันที่มีการรับค่าเข้ามาในฟังก์ชัน และมีการส่งค่ากลับออกไปจากฟังก์ชัน

```
#include<stdio.h>
#include<conio.h>
type func_name(type);
int main()
{
   ...
   var = func_name(varX);
   ...
   return 0;
}
```

```
type func_name(type varY)
{
   statement-1;
   statement-2;
   ...
   statement-n;
   return(varZ);
}
```

ตัวอย่าง ฟังชั่นที่มีรับค่า มีส่งค่า


```
#include<stdio.h>
float  CircleArea( int );
int  main()
{   int  radius;
    printf("input radius : ");    scanf("%d",&radius);
    printf(" Circle area = %f\n",CircleArea(radius));
    return 0;
}
float  CircleArea( int rad )
{   float  answer=0;
    answer  =  22.0/7.0*rad*rad ;
    return  answer;
}
```

ฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ

- เป็นฟังก์ชันที่ไม่มีการรับค่าเข้ามาในฟังก์ชัน แต่มีการส่งค่ากลับออกไปจากฟังก์ชัน

```
#include<stdio.h>
type func_name(void);
int main()
{
   ...
   var = func_name();
   ...
   return 0;
}
```

```
type func_name()
{
   statement-1;
   statement-2;
   ...
   statement-n;
   return(varZ);
}
```

ตัวอย่างฟังก์ชันที่ไม่มีการรับค่า แต่มีการส่งค่ากลับ


```
#include<stdio.h>
int  RoundInput( void );
int  main()
{   int  i,round;
    round = RoundInput();
    for( i=1; i<=round;i++)     printf("hello #%d\n",i);
    return 0;
}
int  RoundInput(void)
{   int answer;
    printf( "Please input number of hello : ");
    scanf( "%d", &answer );
    return  answer;
}
```

ตัวอย่าง

จากโปรแกรมตัวอย่างจงอธิบาย

- ฟังก์ชันชื่ออะไร
- มีการรับค่าหรือไม่ ถ้ามี มีกี่พารามิเตอร์
- ฟังก์ชันรับค่าเป็นตัวแปรชนิดใด
- ฟังก์ชันมีการส่งค่ากลับหรือไม่ ถ้ามี ส่งค่ากลับเป็นตัวแปรชนิดใด
- ถ้าต้องการตัดบรรทัด **/*Function Prototype*/** ออกต้องปรับเปลี่ยนโปรแกรมอย่างไร
- โปรแกรมทำงานอะไร

```c
#include<stdio.h>
#include<math.h>
#define PI 3.14
float deg_rad(float);   /*Function Prototype*/
int main()
{
   float deg,rad;
   printf ("Enter Degree : ");
   scanf ("%f",&deg);
   rad = deg_rad(deg);   //printf ("%f->%f\n",deg,rad);
   printf ("sin(%.2f) = %.3f\n",deg,sin(rad));
   printf ("cos(%.2f) = %.3f\n",deg,cos(rad));
   printf ("tan(%.2f) = %.3f\n",deg,tan(rad));
   return 0;
}
float deg_rad(float num)
{
   float ans;
   ans = num * PI / 180;
   return(ans);
}
```

คำถามท้ายบท

1. จงเขียนโปรแกรมหาพื้นที่ของสามเหลี่ยมใด ๆ โดยรับค่าด้านทั้งสาม(a,b,c)ทางแป้นพิมพ์ จากสูตร

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a+b+c}{2}$$

2. จงเขียนโปรแกรมแลกเปลี่ยนเงินสกุลสหรัฐกับไทย โดยรับค่าเงินดอลล่าร์จากแป้นพิมพ์แล้วแสดงผลเป็นบาท โดยใช้ฟังก์ชั่น D2B(int) เพื่อทำการแปลงดอลลาร์ให้เป็นเงินบาท โดยกำหนดให้ 1 ดอลลาร์มีค่า 31.25 บาท

3. จากส่วนของโปรแกรมต่อไปนี้ เป็นโปรแกรมตรวจสอบตัวเลขที่รับเข้ามาทางแป้นพิมพ์ ว่าเป็นจำนวนเฉพาะหรือไม่ จงเติมส่วนที่หายไปให้ครบ

---→

```
#include<stdio.h>
int main()
{
	int N;
	printf("Input N: ");
	scanf("%d",&N);
	if ( IsPrime(N) )
		printf("%d is Prime number ",N );
	else
		printf("%d is not Prime number ",N );

	return 0;
}
```