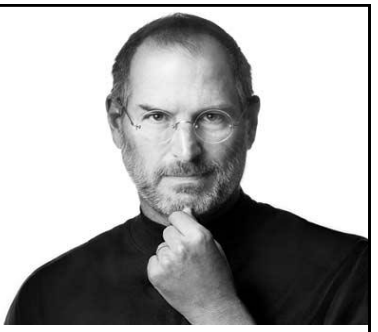
ฟังก์ชันและการประยุกต์



1. การสร้างฟังก์ชัน
2. การประกาศตัวแปร
3. Library function
4. การประยุกต์ฟังก์ชัน
5. ฟังก์ชันเรียกซ้ำ (Recursive function)

ฟังก์ชันแรกในภาษาซี (main)

```
#include “stdio.h”
void main()
{
    int x,y,z;
    x = 100;
    y = 65;
    z = x + y;
    printf(“%d”,z);
}
```

Test.exe

เรียกใช้ฟังก์ชัน test()

```
#include “stdio.h”
void test()
{
    int x, y, z;
    x = 100;
    y = 65;
    z = x + y;
    printf(“%d”,z);
}
void main()
{
    test();
}
```

ฟังก์ชันในภาษาซี

• **ฟังก์ชันมาตรฐาน (Standard Function)**

เป็นฟังก์ชันที่อยู่ในโปรแกรม เก็บไว้ใน header fine ผู้ใช้จะต้องรู้ว่าฟังก์ชันนั้น ๆ เก็บอยู่ในไฟล์ใด

• **ฟังก์ชันที่เขียนขึ้นเอง (user defined function)**

เขียนขึ้นเพื่อให้ทำงานตามที่ต้องการ เช่น

```
void cal_sum()
{
    int x,sum = 0;
    for(x=1; x<=10;x++)
        sum = sum + x;
}
```

ฟังก์ชัน (function)

ฟังก์ชันเป็นกลุ่มคำสั่งที่ใช้ในการแบ่งโปรแกรมเป็นส่วนย่อย ๆ เพื่อทำงานอย่างใดอย่างหนึ่ง ฟังก์ชันนี้อาจมีการส่งข้อมูลไปและกลับระหว่างฟังก์ชันได้

รูปแบบของฟังก์ชัน

จะมีหรือไม่มีขึ้นการลักษณะของฟังก์ชัน

```
ชนิดข้อมูล ชื่อฟังก์ชัน(การประกาศพารามิเตอร์)
{
        การประกาศตัวแปรเฉพาะที่
        กลุ่มคำสั่ง
        [return]
}
```

ค่าที่ส่งกลับ(ถ้ามี)

ประเภทของฟังก์ชันที่สร้างขึ้นเอง

ถ้าหากแยกตามประเภทการส่งค่าไปกลับจะแยกได้ดังนี้

เช่นให้หาค่าที่เข้ามาทางอินพุต

- ฟังก์ชันที่ไม่มีการส่งค่าไปและรับค่ากลับ
- ฟังก์ชันที่มีการส่งค่าไปแต่ไม่มีการรับค่ากลับ

เช่น สั่งให้พิมพ์ข้อความ

- ฟังก์ชันที่มีการส่งค่าไปและรับค่ากลับ

ตัวอย่างเช่นส่งค่าไปให้คำนวณ

ฟังก์ชันที่ไม่มีทั้งการส่งค่าไปและรับค่ากลับ

ฟังก์ชันประเภทนี้จะให้ชนิดของข้อมูลเป็น void และภายในวงเล็บใช้ void เพื่อบอกว่าไม่มีการส่งและรับค่า และจะไม่ใช้คำสั่ง return

รูปแบบของฟังก์ชัน

```
void  funct_name(void)
{
        local variable declaration;
        statement();
}
```

ฟังก์ชันที่มีการส่งค่าไปแต่ไม่มีการรับกลับ

ฟังก์ชันประเภทนี้จะให้ชนิดของข้อมูลเป็น void ภายในวงเล็บจะมีอาร์กิวเมนต์ และจะไม่ใช้คำสั่ง return

```
void ADD(int a, int b)
{
    int x;
    x = a+b;
    printf("sum = %d",x);
}
```

รูปแบบของฟังก์ชัน

```
void  funct_name(type1 arg1, type2 arg2,……,typeN argN)
{
        local variable declaration;
        statement();
}
```

ฟังก์ชันที่มีการส่งค่าไปและมีการรับกลับ

ฟังก์ชันประเภทนี้จะให้ชนิดของข้อมูลเป็นประเภทที่จะส่งกลับ ภายในวงเล็บจะมีอาร์กิวเมนต์ และจะใช้คำสั่ง return ส่งค่ากลับมาให้ฟังก์ชัน

```
int ADD(int a, int b)
{
    int x;
    x = a+b;
    return x;
}
```

รูปแบบของฟังก์ชัน

```
type  funct_name(type1 arg1, type2 arg2,……,typeN argN)
{
        local variable declaration;
        statement();
        return (value);
}
```

ข้อมูลประเภทเดียวกัน

ตัวอย่างโปรแกรม สร้างฟังก์ชันมาสองฟังก์ชัน ไม่มีการส่งและรับค่า

```
#include <stdio.h>
void one(void);
void two(void);
void main(void)
{
      clrscr( );
      one( );
      two( );
}
```

มีการประกาศ Prototype

```
void one( )
{
      int a = 5, b = 7;
      printf("A = %d, B = %d\n",a,b);
}
void two( )
{
     float p = 4.5, q = 3.5;
     printf("P = %6.2f, Q = %6.2f\n"
            ,p,q);
}
```

จะสังเกตเห็นว่าไม่มีพารามิเตอร์ ไม่มี return

ตัวอย่างโปรแกรม สร้างฟังก์ชันมาสองฟังก์ชัน ไม่มีการส่งและรับค่า อีกแบบ

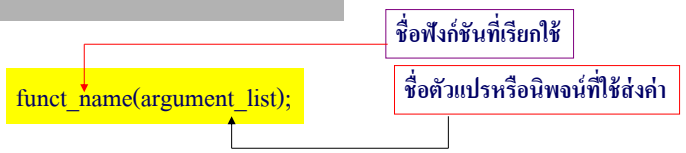
```
#include <stdio.h>
void one( )
{
      int a = 5, b = 7;
      printf("A = %d, B = %d\n",a,b);
}
void two( )
{
     float p = 4.5, q = 3.5;
    printf("P = %6.2f, Q = %6.2f\n",p,q);
}
```

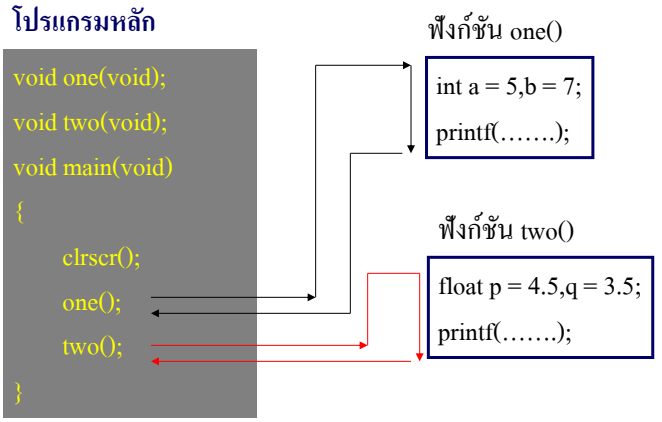
```
void main(void )
{
      clrscr( );
      one( );
      two( );
}
```

ถ้าหากสร้างฟังก์ชันก่อน main จะไม่ประกาศ Prototype ก็ได้

การเรียกใช้ฟังก์ชัน

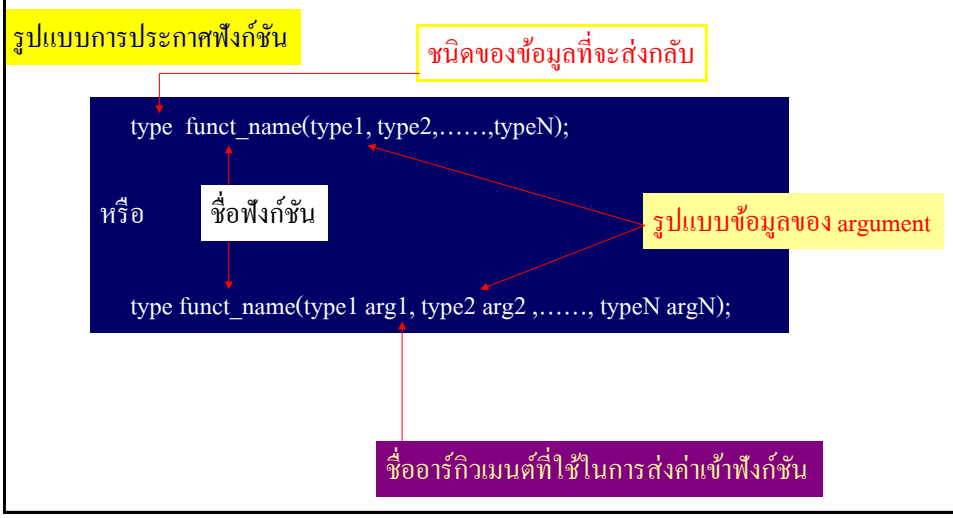
รูปแบบ





การประกาศรูปแบบฟังก์ชัน (function prototype)

คล้ายกับการประกาศตัวแปร เพื่อบอกว่าในโปรแกรมของเรามีฟังก์ชันอะไรที่สร้างขึ้นเอง ฟังก์ชันนั้น ๆ มีการเรียกใช้อย่างไร ใช้ตัวแปรอย่างไร ตำแหน่งที่ประกาศ **ควรจะอยู่ก่อน main()**



ตัวอย่างการประกาศรูปแบบฟังก์ชัน

ตัวอย่าง

1. float add(int, float);
2. int sum(int p, int q); หรือ int sum(int, int);
3. void move(int, float, int);
4. float calculate(float, float);
5. void point(int*, int*, float);

ฟังก์ชันนี้ใช้อาร์กิวเมนต์ที่เป็นตัวแปรแบบพอยน์เตอร์

ฟังก์ชันนี้ไม่มีการคืนค่า จะรับอาร์กิวเมนต์เป็น int, float และ int เช่น move(5, 2.70, 12);

การประกาศตัวแปรของฟังก์ชัน

ตัวแปรแบบทั่วไป (global variables)

เป็นตัวแปรที่ทุกส่วนของโปรแกรมสามารถเรียกใช้ได้ บางครั้งจะเรียกว่าตัวแปรเอ็กเทอร์นัล (External variables)

ตัวแปรเฉพาะที่ (local variables)

เป็นตัวแปรที่สร้างขึ้นภายในฟังก์ชัน ใช้ในฟังก์ชันนั้น ๆ เมื่อออกจากฟังก์ชันค่าตัวแปรจะหายไป การใช้ตัวแปรประเภทนี้จะทำให้ฟังก์ชันต่าง ๆ สามารถใช้ชื่อตัวแปรชื่อเดียวกันได้ ตัวแปรประเภทนี้บางครั้งเรียกว่า ออโตเมติก(automatic variable)

ตัวอย่างการใช้ตัวแปรแบบทั่วไปร่วมกับฟังก์ชัน

```
#include <stdio.h>
int a;
void Ex()
{
      a = 5;
      printf("%d\n",a);
}
```

```
main()
{
      a = 3;
      printf("%d\n",a);
      Ex();
      printf("%d\n",a);
       return  0;
}
```

3
5
5

ตัวอย่างการใช้ตัวแปรเฉพาะที่

```
#include <stdio.h>
int a;
void Ex()
{
      int  a;
      a = 5;
      printf("%d\n",a);
}
```

```
main()
{
      a = 3;
      printf("%d\n",a);
      Ex();
      printf("%d\n",a);
       return  0;
}
```

3
5
3

ตัวอย่างฟังก์ชันที่คืนค่าเป็นเลขจำนวนเต็ม

```
#include <stdio.h>
int x,y;
int add_num(int a, int b)
{
      int  m;
      m = a + b;
      return m;
}
```

```
void main()
{
      printf("INPUT   X ");
      scanf("%d",&x);
      printf("INPUT   Y ");
      scanf("%d",&y);
      printf("%d + %d = %d\n",
      x,y,add_num(x,y) );
}
```

ส่งตัวแปร x,y ไปให้ a กับ b

ตัวอย่างการฟังก์ชันที่ส่งผ่านตัวแปรหลายประเภท

```
#include <stdio.h>
void test_f(int a, int b, double f, char c, char s[20])
{
      printf("Integer %d, %d\n",a, b);
      printf("Float  %f\n", f);
      printf("Char  %c\n", c);
      printf("String %s\n",s);
}
void main()
{
      test_f(23, 34, 3.564, 'p' ,Yellow River");
}
```

ผลการทำโปรแกรม

```
Integer  23, 34
Float  3.564000
Char  p
String Yellow River
```

ฟังก์ชันมาตรฐาน

ฟังก์ชันทางคณิตศาสตร์ #include <math.h>

ฟังก์ชัน sin(x), cos(x), tan(x)

หาค่าของฟังก์ชันโดยรับค่ามุมเป็นเรเดียน

ฟังก์ชัน sqrt(x)

หาค่ารากที่สองของ x

ต้องรู้ว่ารับข้อมูลเข้าเป็นอะไร และคืนค่าประเภทใดออกมา

ฟังก์ชัน exp(x)

ฟังก์ชันหาค่า e^x โดย e มีค่าประมาณ 2.718282

ฟังก์ชัน pow(x,y)

ใช้หาค่า x^y โดย x เป็นค่าคงที่หรือตัวแปรที่มีค่ามากกว่า 0 y เป็นค่ายกกำลัง

ฟังก์ชัน lox(x) , log10(x)

หาค่า log ฐาน n และค่า log ฐาน 10

โปรแกรมหาค่าจากฟังก์ชันตรีโกณมิติ

```
#include “stdio.h”
#include “math.h”
main()
{
      double r, pi = 3.14159;
      r = pi/180;   clrscr();
      printf(“%f\n”,sin(r));
      printf(“%f\n”,cos(r));
      printf(“%f\n”,tan(r));
}
```

อาจใช้ M_PI

360 องศา = 2*pi

x องศา = (pi/180)*x

ผลการรัน

0.017452
0.999848
0.017455

ฟังก์ชันที่พบบ่อย

ฟังก์ชัน clrscr()

ใช้ลบจอภาพ

ฟังก์ชัน gotoxy(x,y)

กำหนดตำแหน่งของเคอร์เซอร์

ฟังก์ชัน abs(x)

หาค่าสัมบูรณ์ของ x

ฟังก์ชัน atoi(s)

เปลี่ยนค่าสตริง s ให้เป็นเลขจำนวนเต็ม

ฟังก์ชัน atof(s)

เปลี่ยนค่าสตริง s ให้เป็นเลขทศนิยม

ตัวอย่างการหาค่าของ $f(x) = x^2 + 3x + 1$

```
#include <stdio.h>
int f_x(int x);
main()
{
      int i;
      printf(" x      x^2 + 3x + 1  \n");
      for(i = 1; i <= 10; i++)
            printf("%d      %d \n",i,f_x(i) );
      getch();
}
```

```
int f_x(int x)
{
      int y;
      y = x*x + 3*x + 1;
       return  y;
}
```

ตัวอย่าง

แสดงการทำงานของโปรแกรมการบวกเลขจำนวนจริง 2 จำนวนที่รับจากผู้ใช้

```
#include <stdio.h>
double  InputDouble ( )
{
        double  x;
        printf ( “\nInput real value : “ );
        scanf ( “%.2f  ”, &x );
        return ( x );
}
```

23

ตัวอย่าง (ต่อ)

```
double  SumDouble ( double  x,  double  y )
{
        return ( x + y );
}
void  PrintOut ( double  x )
{
        printf ( “\n Result of sum is : %.2f”,  x );
}
```

24

ตัวอย่าง (ต่อ)

```
void  main (  )
{
        double   a1, a2, sumVal;
        a1  =  InputDouble( );
        a2  =  InputDouble( );
        sumVal  =  SumDouble ( a1, a2 );
        PrintOut ( sumVal );
}
```

25

ตัวอย่าง

แสดงการทำงานของโปรแกรมการบวกเลขจำนวนจริง 2 จำนวนที่รับจากผู้ใช้ ในลักษณะที่มีการประกาศโปรโตไทป์

```
#include <stdio.h>
double  InputDouble ( );
double  SumDouble ( double ,  double  );
void     PrintOut ( double );
```

26

ตัวอย่าง (ต่อ)

```
void main ( )
{
        double  a1, a2, sumVal;
        a1 = InputDouble( );
        a2 = InputDouble( );
        sumVal = SumDouble ( a1, a2 );
        PrintOut ( sumVal );
}
```

27

4.3 การเรียกใช้ฟังก์ชัน

การเรียกใช้ฟังก์ชันที่มีการคืนค่า จะใช้รูปแบบดังต่อไปนี้

ค่าที่รับ = ฟังก์ชัน (อาร์กิวเมนต์)

28

ตัวอย่าง

a1 ต้องมีชนิดเป็น double เนื่องจากค่าที่จะส่งคืนกลับมาจากฟังก์ชันมีชนิดเป็น double

```
a1  =  InputDouble ( );
```

ใช้คู่กับโปรโตไทป์

```
double  InputDouble ( );
```

29

ตัวอย่าง

a1 และ a2 ต้องมีชนิดเป็น double เพื่อให้ตรงกับชนิดตัวแปรของอาร์กิวเมนท์ที่ประกาศในโปรโตไทป์

```
sumVal =  SumDouble (a1,a2 );
```

ใช้คู่กับโปรโตไทป์

```
double  InputDouble ( );
```

30

ตัวอย่าง

```
PrintOut( sumVal );
```

ใช้คู่กับโปรโตไทป์

```
void PrintOut ( double );
```

ประกาศให้รู้ว่าฟังก์ชันนี้ไม่มีการคืนค่า

31

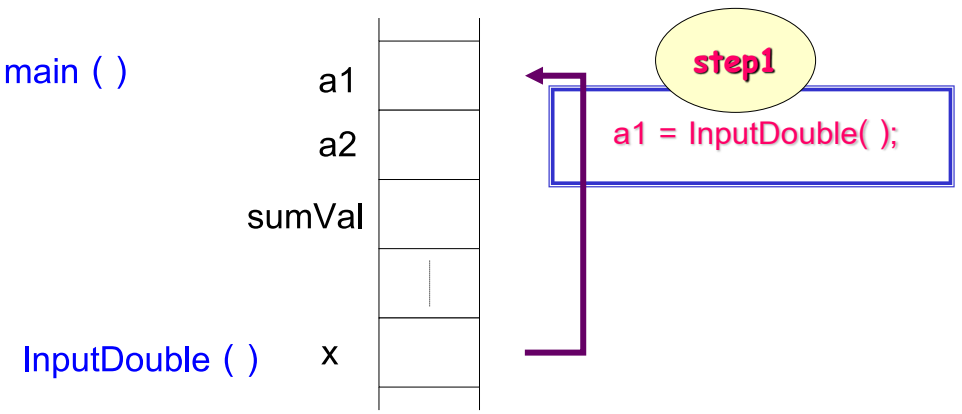
ขอบเขต (Scope)

การทำงานของโปรแกรมภาษาซีจะทำงานที่ฟังก์ชัน main ( ) ก่อนเสมอ เมื่อฟังก์ชัน main ( ) เรียกใช้งานฟังก์ชันอื่น ก็จะมีการส่งคอนโทรล (Control) ที่ควบคุมการทำงานไปยังฟังก์ชันนั้น ๆ จนกว่าจะจบฟังก์ชัน หรือพบคำสั่ง return

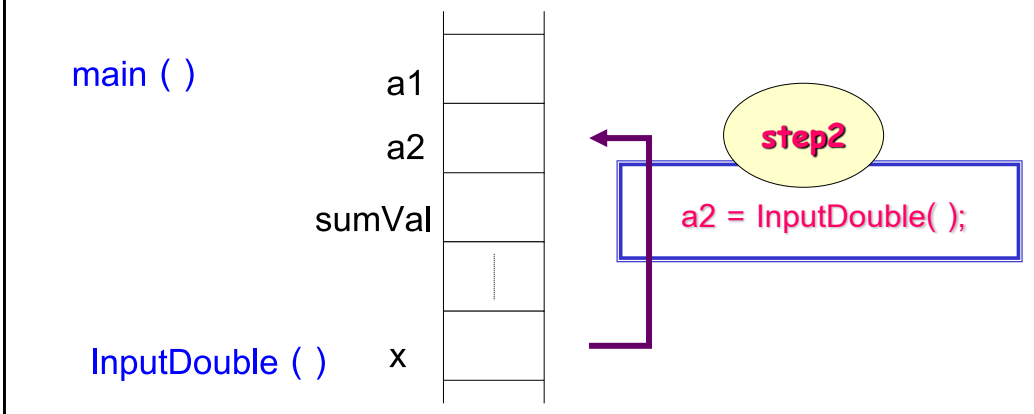
32

เมื่อมีการเรียกใช้งานฟังก์ชันจะมีการจองพื้นที่หน่วยความจำสำหรับตัวแปรที่ต้องใช้ภายในฟังก์ชันนั้น และเมื่อสิ้นสุดการทำงานของฟังก์ชันก็จะมีการคืนพื้นที่หน่วยความจำส่วนนั้นกลับสู่ระบบ การใช้งานตัวแปรแต่ละตัวจะมีขอบเขตของการใช้งานขึ้นอยู่กับตำแหน่งที่ประกาศตัวแปรนั้น

ตัวอย่าง จากตัวอย่างที่ผ่านมา สามารถแสดงขอบเขตการทำงานได้ดังนี้

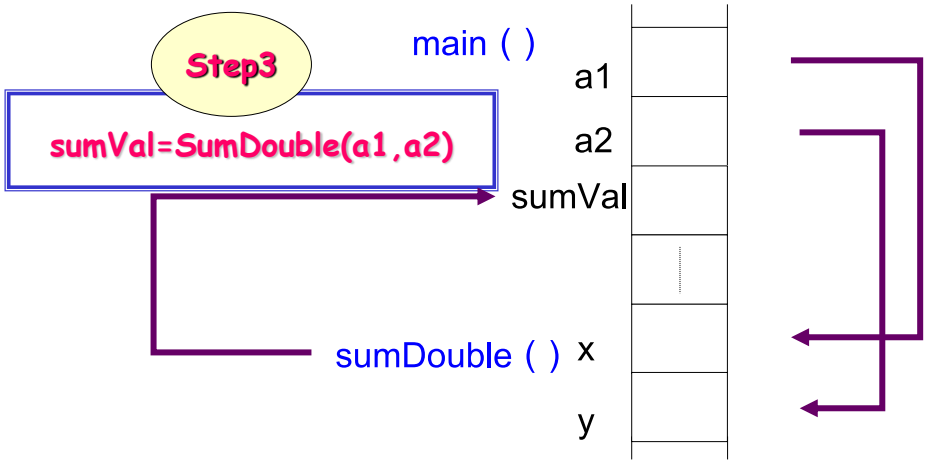


ตัวอย่าง (ต่อ)



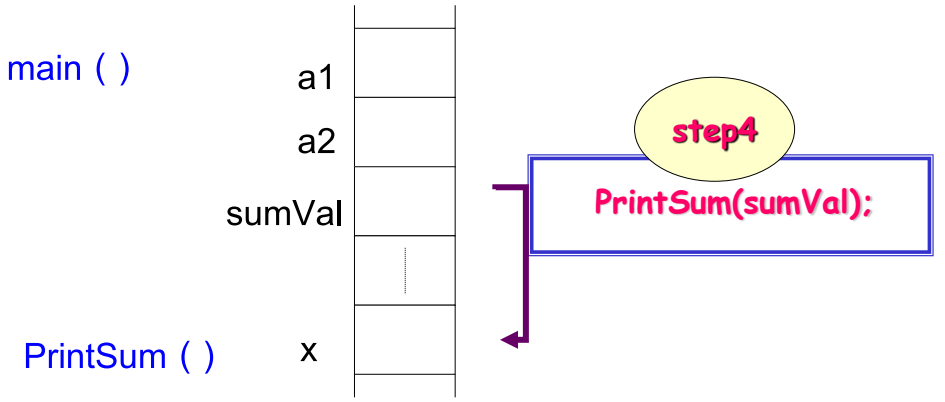
35

ตัวอย่าง (ต่อ)



36

ตัวอย่าง (ต่อ)



37

จะเห็นว่าตัวแปร x ที่ประกาศในแต่ละขั้นตอนจะทำงานอยู่ภายในฟังก์ชันที่มีการประกาศค่าเท่านั้น และใช้พื้นที่ในการเก็บข้อมูลคนละส่วนกัน

ขอบเขตการทำงานของตัวแปรแต่ละตัวจะกำหนดอยู่ภายบล็อกของคำสั่งภายในเครื่องหมายปีกกา ({ }) หรือการประกาศในช่วงของการประกาศฟังก์ชัน เรียกตัวแปรเหล่านี้ว่า

ตัวแปรโลคอล (Local Variable)

38

นอกจากนี้สามารถประกาศตัวแปรไว้ที่ภายนอกฟังก์ชันบริเวณส่วนเริ่มของโปรแกรมจะเรียกว่า ตัวแปรโกลบอล (Global Variable) ซึ่งเป็นตัวแปรที่สามารถเรียกใช้ที่ตำแหน่งใด ๆ ในโปรแกรมก็ได้ ยกเว้นในกรณีที่มีการประกาศตัวแปรที่มีชื่อเดียวกันตัวแปรโกลบอลภายในบล็อกหรือฟังก์ชัน

39

ลองเขียนโปรแกรมวนลูปรับตัวเลข แล้วคืนค่าเป็นตัวเลข

```
char   ch;
ch  =  getch() – ‘0’;
```

ค่าคงที่ของฟังก์ชันทางคณิตศาสตร์

สัญลักษณ์	ความหมาย
M_E	e
M_LOG2E	log e
M_LOG10E	log(10) e
M_LN2	ln 2
M_LN10	ln 10
M_PI	pi
M_PI_2	pi/2

ฟังก์ชันเรียกซ้ำ (Recursive Function)

เป็นฟังก์ชันที่สร้างขึ้นเอง โดยภายในฟังก์ชันจะมีการเรียกชื่อของตัวเอง บางครั้งจะเรียกว่าฟังก์ชันแบบเรียกตัวเอง โปรแกรมประเภทนี้จะต้องมีจุดที่ให้ออกจากฟังก์ชันได้

ตัวอย่าง

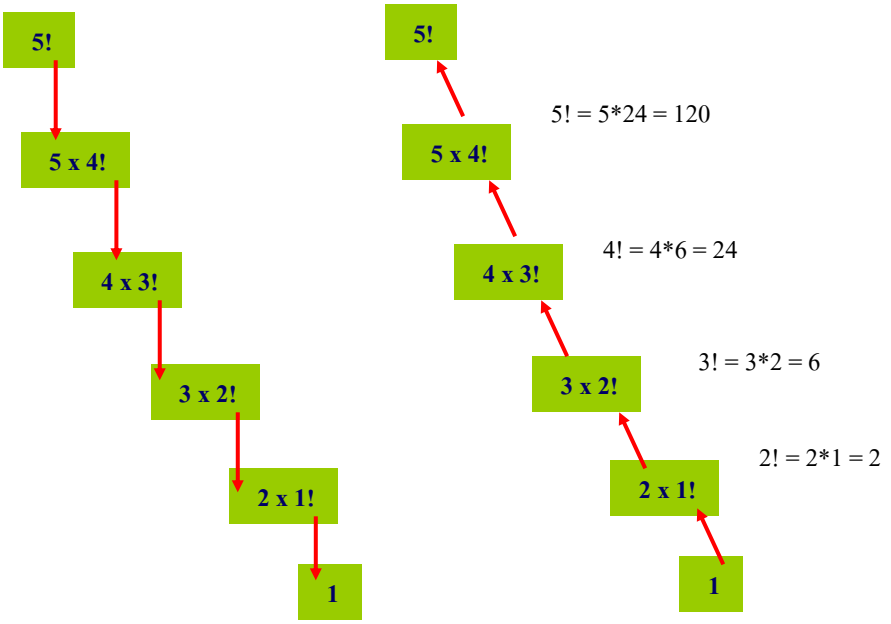
5! = 5 x 4 x 3 x 2 x 1

สามารถมองได้เป็น

5! = 5 x 4 !

N! = N x (N-1)!

การคืนค่าของฟังก์ชัน



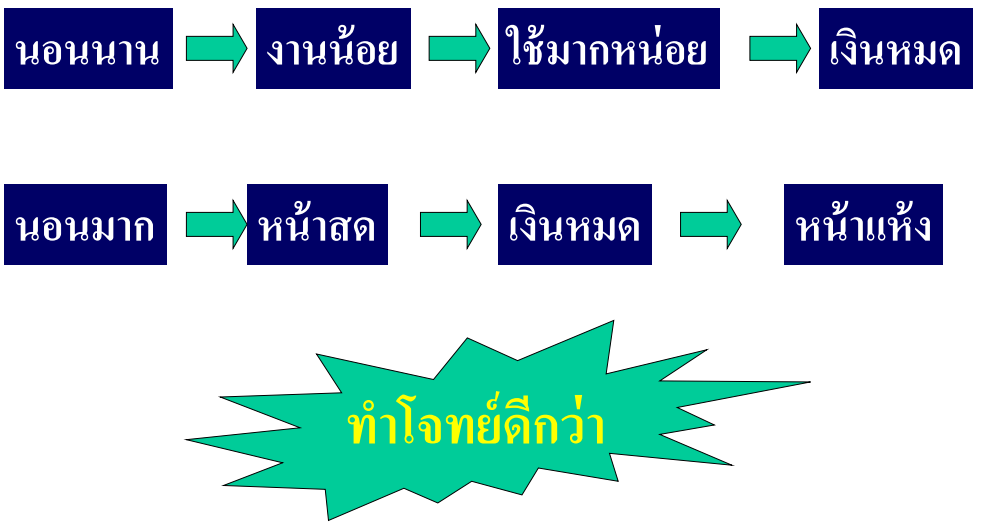
ตัวอย่างการฟังก์ชันหาค่า factorial

```
#include <stdio.h>
long factorial(long)
main()
{
      int i;
      for(i = 1 ; i<= 10; i++)
            printf("%2d! = %ld\n",i,factorial(i));
}
```

```
long factorial(long number)
{
    if (number <= 1)
        return  1;
    else
      return(number*factorial(number-1));
}
```

ฟังก์ชันสุ่มตัวเลข

```
#include "stdlib.h"
#include "time.h"
main()
{
    srand(time(NULL));
    printf("%d\n",rand()%10);
    getch();
}
```



แบบฝึกหัดเพิ่มเติม

1. จงเขียนฟังก์ชั่นตรวจสอบว่ากด y/Y หรือ n/N หรือไม่
2. เขียนฟังก์ชันรับข้อมูล 10 ค่าแล้วหาผลรวม
3. เขียนโปรแกรมหาค่า sin, cos, tan โดยมีมุมเป็นองศา ตั้งแต่ 0 ถึง 90 โดยให้กระโดดครั้งละ 5
4. เขียนโปรแกรมหาค่า $f(x) = x^2 + 1$
5. เขียนโปรแกรมหาพื้นที่ใต้กราฟของ $y(x) = x^2 - 3x + 2$