บทที่ 9 ฟังก์ชัน (Function)

รายวิชา สธ 113 การออกแบบโปรแกรมทางธุรกิจเบื้องต้น

อ.อภิพงศ์ ปิงยศ

Outline

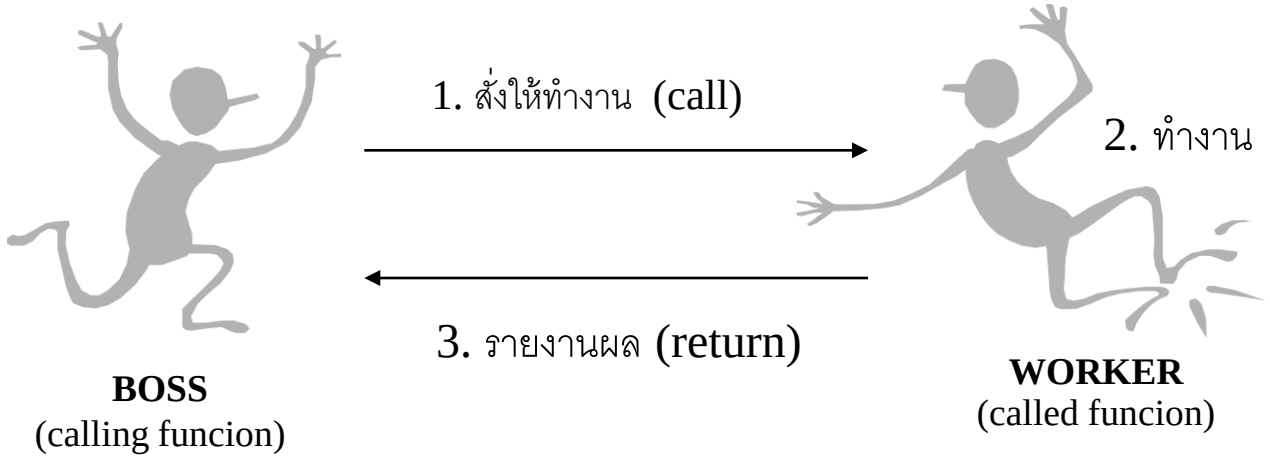
- บทนำ
- โปรแกรมย่อยภาษาซี
- การนิยามฟังก์ชัน
- ต้นแบบของฟังก์ชัน (Prototype)
- ตัวแปรภายใน และตัวแปรภายนอก

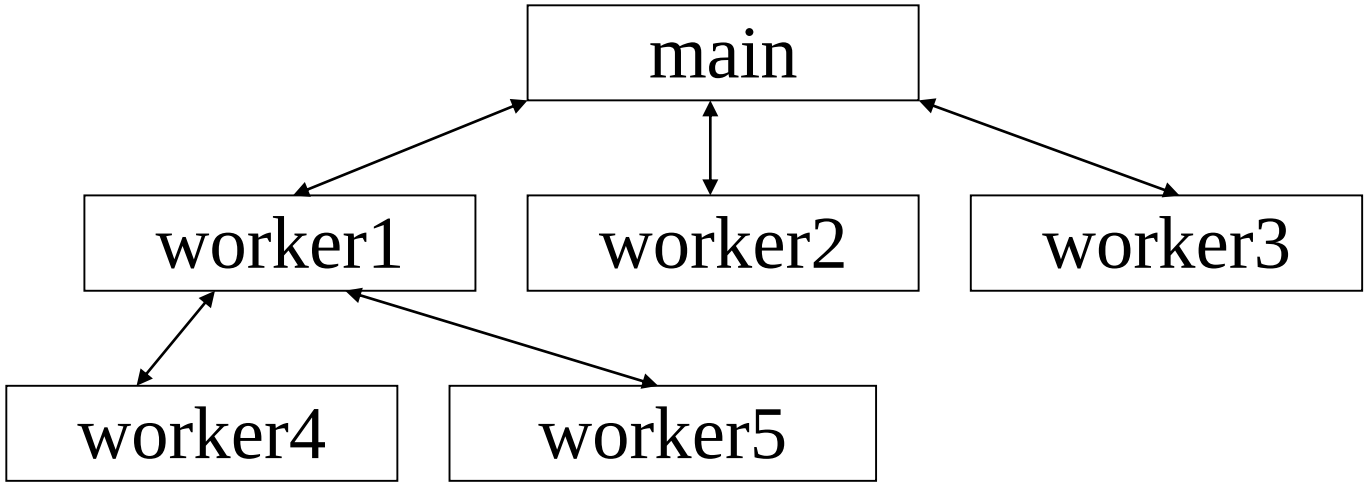
บทนำ

- โปรแกรมมีขนาดใหญ่ ทำให้การเขียนไว้ภายใต้ main() เกิดความยุ่งยาก ผิดพลาดง่าย นอกจากนั้นเมื่อต้องการตรวจสอบความผิดพลาดที่เกิดจากโปรแกรมยังทำได้ยากอีกด้วย

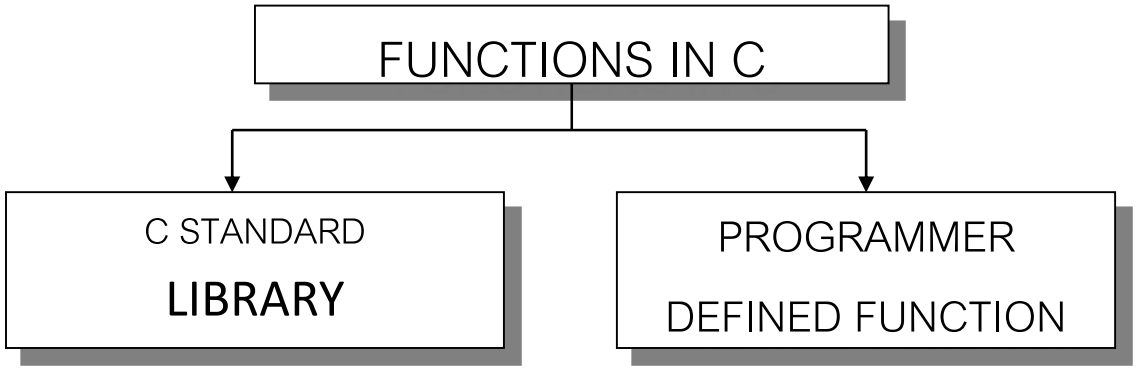
- Divide and Conquer คือการแก้ไขปัญหาข้างต้น โดยแบ่งโปรแกรมออกเป็นส่วนย่อย ๆ ที่มีขนาดเล็ก (Module) โดยเมื่อสร้างและทดสอบโปรแกรมย่อย ๆ แล้ว ก็ประกอบเป็นโปรแกรมใหญ่ที่สมบูรณ์ในขั้นตอนสุดท้าย

โปรแกรมย่อย ในภาษาซี





โปรแกรมย่อย ในภาษาซี



- ฟังก์ชันมาตรฐานในภาษาซี (C Standard Function)
  - ฟังก์ชันซึ่งอยู่ในไลบรารีภาษาซีมาตรฐาน (C Standard Library) ซึ่งประกอบไปด้วยฟังก์ชันพื้นฐานที่จำเป็นสำหรับการทำงาน เช่น การจัดการกับ input/output (printf ซึ่งอยู่ในไลบรารี stdio) การคำนวณทางคณิตศาสตร์ (sqrt ซึ่งอยู่ในไลบรารี math)

printf() ⟶ **#include <stdio.h>**

โปรแกรมย่อย ในภาษาซี

- ฟังก์ชันที่สร้างขึ้นใหม่โดยโปรแกรมเมอร์ (Programmer-Defined Function)
  - โปรแกรมเมอร์สามารถเขียนฟังก์ชันเพื่อนิยามการทำงานที่จะเรียกใช้ในส่วนต่างๆของโปรแกรม โดยฟังก์ชันการทำงานดังกล่าวจะถูกเขียนไว้ในฟังก์ชันเพียงครั้งเดียวเท่านั้น แต่สามารถเรียกใช้งานได้หลายครั้ง

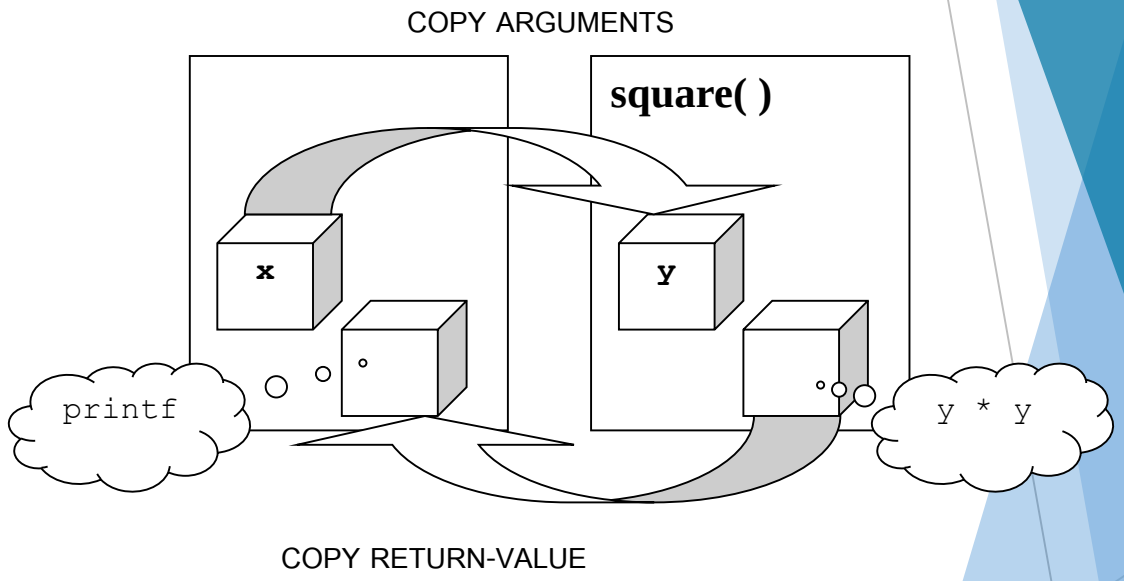
การนิยามฟังก์ชัน

```
#include<stdio.h>

int main( )
{
        int i;

        for(i = 1; i <= 10; i++)
            printf("%d ", i * i);

        printf("\n”);
        return 0;
}
```

```
#include<stdio.h>

int square(int);

int main( )
{
        int i;

        for(i = 1; i <= 10; i++)
            printf("%d ", square(i));

        printf("\n”);
        return 0;
}

int square(int y)
{
        return(y*y);
}
```

การนิยามฟังก์ชัน

```
#include<stdio.h>

int square(int);

int main( )
{
        int i;

        for(i = 1; i <= 10; i++)
                printf("%d ", square(i));

        printf("\n”);
        return 0;
}

int square(int y)
{
        return(y*y);
}
```



การนิยามฟังก์ชัน

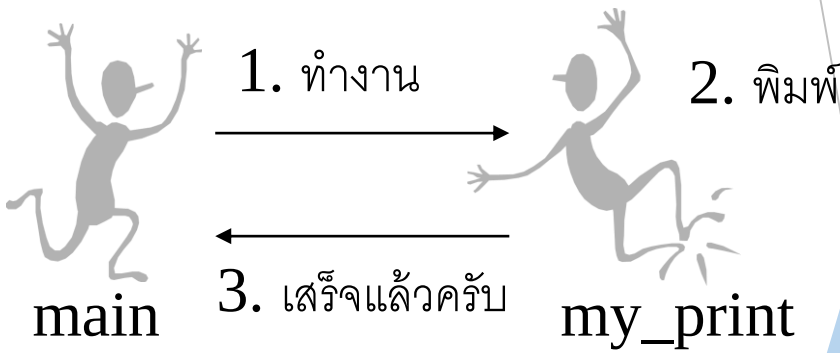
```
return-value-type function-name(parameter-list)
{
      declarations;
      statements;
}
```

- return-type-value คือ ชนิดของข้อมูลที่ต้องการจะส่งค่าผ่านกลับออกไปยังฟังก์ชันผู้เรียกใช้หรือจุดที่เรียกใช้งานเช่น int หรือ float หากไม่ส่งค่ากลับให้ใช้ void
- function-name คือ ชื่อของฟังก์ชันที่จะถูกสร้างขึ้น
- parameter-list คือรายการตัวแปรพารามิเตอร์ทั้งหมดที่ต้องการส่งผ่านไปยังฟังก์ชันเมื่อเรียกใช้ ตามลำดับ
- declarations คือ ส่วนของการประกาศตัวแปรภายในฟังก์ชัน
- statements คือ ส่วนของคำสั่งที่กระทำในฟังก์ชันนั้น

รูปแบบที่ 1

ตัวอย่างที่ 1: ฟังก์ชันที่ไม่รับผ่านค่าตัวแปร และไม่ส่งผ่านค่ากลับ

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

void my_print()
{
        printf(“Hello world”);
}
```

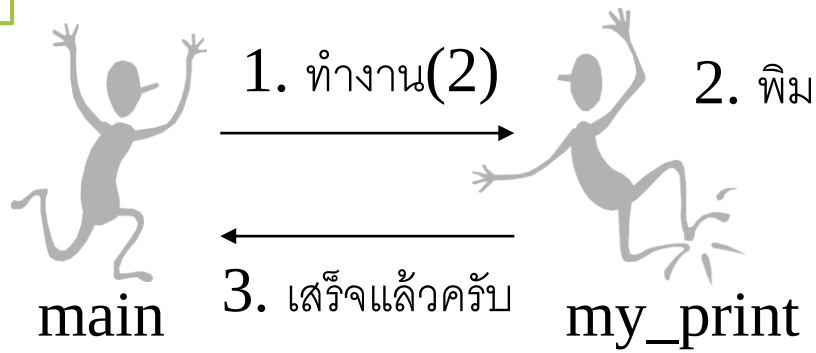


รูปแบบที่ 2

ตัวอย่างที่ 2.1: ฟังก์ชันที่มีการรับผ่านค่าตัวแปร แต่ไม่ส่งผ่านค่ากลับ

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

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

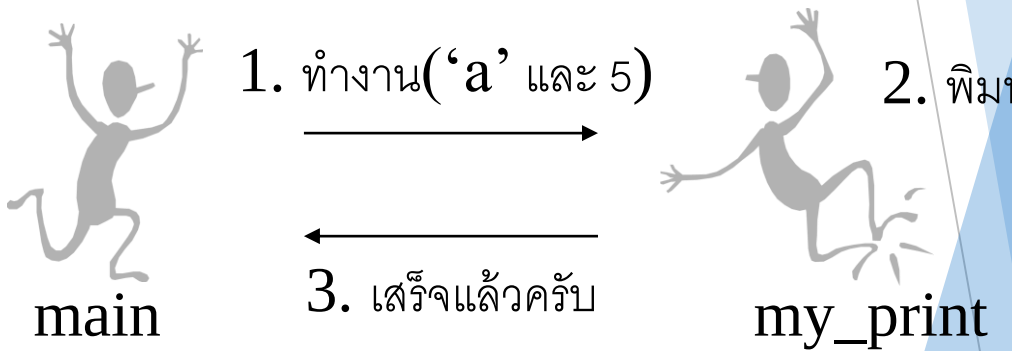
Argument



รูปแบบที่ 2

ตัวอย่างที่ 2.2: ฟังก์ชันที่มีการรับผ่านค่าตัวแปร แต่ไม่ส่งผ่านค่ากลับ

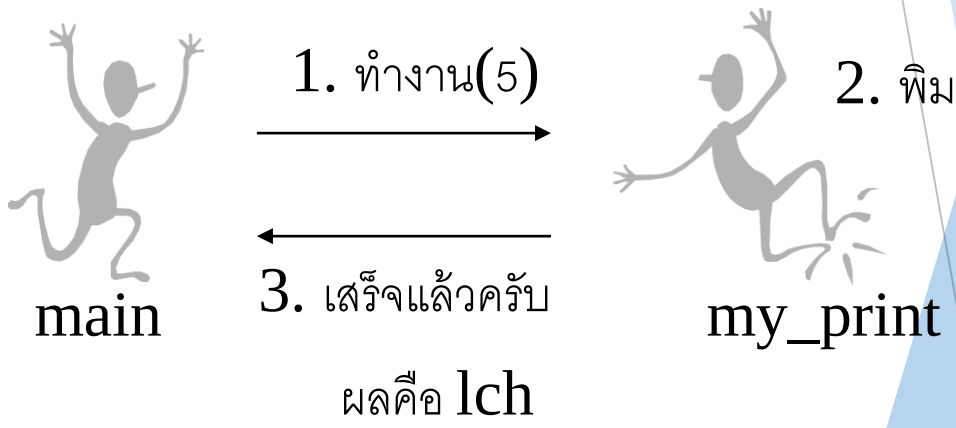
```
void main()
{
        my_print('a', 5);
}

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



รูปแบบที่ 3

ตัวอย่างที่ 3: ฟังก์ชันที่มีทั้งการรับผ่านค่า และส่งผ่านค่ากลับ

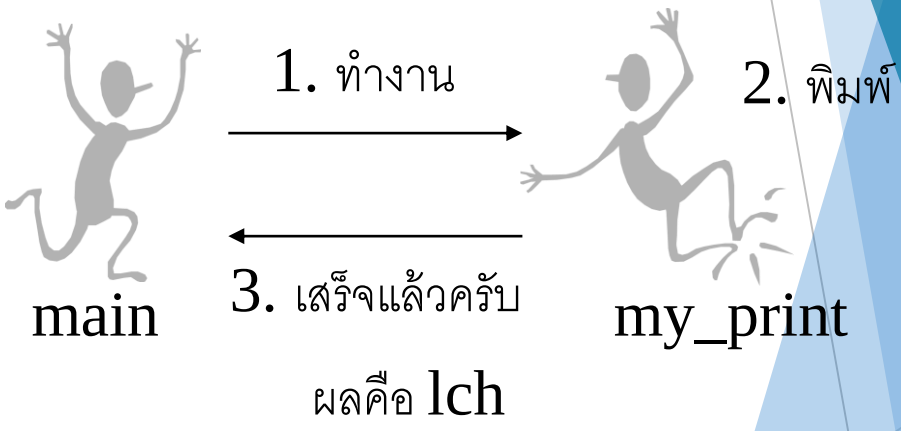
```
void main()
{
    char ch;
    ch = my_print(5);
    printf("%c\n", ch);
}

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



รูปแบบที่ 4

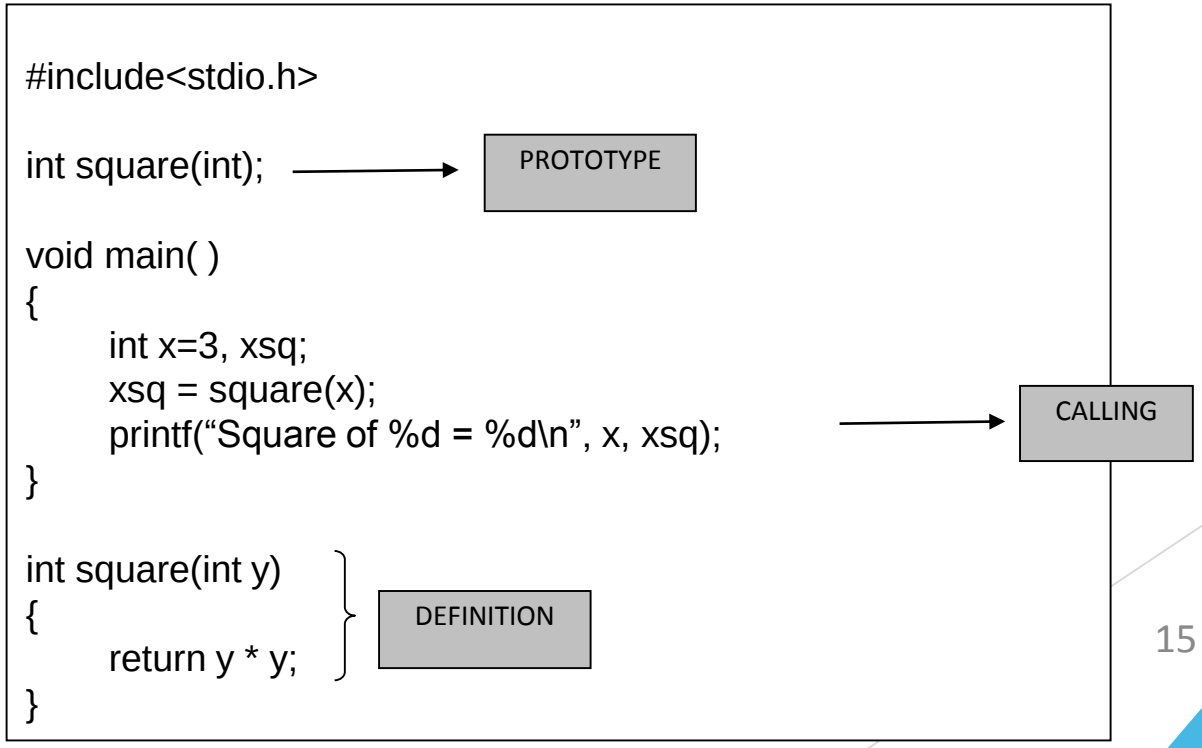
ตัวอย่างที่ 4: ฟังก์ชันที่ไม่มีการรับผ่านค่า และแต่มีการส่งผ่านค่ากลับ

```
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;
}
```



ต้นแบบของฟังก์ชัน (Prototype)

ต้นแบบของฟังก์ชันเป็นตัวบอกให้ตัวแปลภาษารู้ถึงชนิดของข้อมูลที่จะส่งค่ากลับ จำนวนของตัวแปรพารามิเตอร์ที่ฟังก์ชันคาดหวังว่าจะได้รับ ชนิดของพารามิเตอร์แต่ละตัว และลำดับของพารามิเตอร์เหล่านั้น ตัวแปลภาษาสามารถที่จะนำข้อมูลเหล่านี้ในการตรวจสอบความถูกต้องของการเรียกใช้ฟังก์ชัน ในตัวอย่าง int square(int); คือต้นแบบของฟังก์ชัน



ต้นแบบของฟังก์ชัน

```
#include <stdio.h>

int maximum(int, int, int); //function prototype

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;
}

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

โปรแกรมเรียกใช้ฟังก์ชันก่อนการนิยาม ดังนั้นเพื่อการตรวจสอบความถูกต้องในการเรียกใช้ จึงจำเป็นต้องมีต้นแบบของฟังก์ชัน

```
int maximum(int, int, int);
```

ต้นแบบของฟังก์ชัน (ตัวอย่างที่สามารถเขียนได้)

```
#include <stdio.h>

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

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;
}
```

โปรแกรมได้นิยามฟังก์ชันไว้ก่อนการเรียกใช้ จึงไม่จำเป็นต้องมีต้นแบบฟังก์ชัน แต่ข้อเสียคือฟังก์ชัน main จะถูกเลื่อนไปท้ายโปรแกรม เนื่องจากฟังก์ชัน main เป็นฟังก์ชันหลัก ในการอ่านโปรแกรมจึงควรพบฟังก์ชันนี้ก่อน นอกจากนี้การเขียนในลักษณะนี้ยังไม่เหมาะสมในกรณีที่โปรแกรมมีขนาดใหญ่มากและซับซ้อนด้วย

ตัวแปรภายใน และตัวแปรภายนอก

ตัวแปรภายใน (Local Variable) คือ ตัวแปรที่ถูกสร้างขึ้นภายในฟังก์ชัน สามารถเรียกใช้งานได้เฉพาะภายในฟังก์ชันที่สร้างขึ้น และจะถูกทำลายลงเมื่อเสร็จสิ้นการทำงานของฟังก์ชันนั้นๆ

ตัวแปรภายนอก (Global Variable) คือ ตัวแปรที่ถูกสร้างขึ้นภายนอกฟังก์ชัน สามารถใช้งานได้ในทุกฟังก์ชัน หรือทั้งโปรแกรม (ยกเว้นฟังก์ชันที่มีตัวแปรภายในชื่อเดียวกับตัวแปรภายนอก ซึ่งมีความเป็นไปได้) และจะคงอยู่ตลอดการทำงานของโปรแกรม

ตัวแปรภายใน และตัวแปรภายนอก

```
#include <stdio.h>

int ans = 0;

/* function prototype */
int inc_one(int);

void main()
{
    ans = inc_one(3);
    printf("Answer is %d\n", ans);

}

int inc_one(int x)
{
    int ans, b;
    b = 2;
    ans = x + b;
    return ans;
}
```

ans เป็นตัวแปรภายนอกเพราะประกาศไว้ข้างนอก ซึ่งจะเห็นว่าฟังก์ชัน main สามารถเรียกใช้ได้โดยไม่ต้องประกาศตัวแปร ans อีก

x, ans, b เป็นตัวแปรภายในฟังก์ชัน inc_one ที่รู้จักแต่ในฟังก์ชันนี้เท่านั้น และตัวแปร ans ที่ประกาศในฟังก์ชันนี้จะเป็นคนละตัวกับตัวแปร ans ที่ประกาศข้างนอกไม่เกี่ยวข้องกัน

ตัวแปรภายใน และตัวแปรภายนอก

```
#include <stdio.h>

/* prototype of my_func */
void my_func();

void main()
{
   int x=3;
   printf("Main: Before call function x=%d\n", x);
   my_func();          /* call my_func */
   printf("Main: After call function x=%d\n", x);
}

void my_func()
{
   int x;
   x=2;
   printf("My_func: x=%d\n", x);
}
```

```
Main: Before call function x=3
My_func: x=2
Main: After call function x=3
```

ตัวอย่างนี้แสดงถึงตัวแปรภายในฟังก์ชัน main ชื่อ x และตัวแปรภายในฟังก์ชัน my_func ชื่อ x ซึ่งแม้จะมีชื่อเดียวกันแต่ค่าที่เก็บก็ไม่เกี่ยวข้องกัน ดังนั้นแม้เมื่อฟังก์ชัน my_func ถูกเรียกใช้ซึ่งค่าในตัวแปร x ในฟังก์ชัน my_func ถูกกำหนดเป็น 2 ค่า x ในฟังก์ชัน main ก็ยังมีค่าเป็น 3 เหมือนเดิม ดังผลการทำงานของโปรแกรม

ตัวแปรภายใน และตัวแปรภายนอก

```
#include <stdio.h>

void my_func(); /* prototype of my_func */

int x;

void main()
{
   printf("Main: Before call function x=%d\n", x);
   my_func();          /* call my_func */
   printf("Main: After call function x=%d\n", x);
}

void my_func()
{
   x=2;
   printf("My_func: x=%d\n", x);
}
```

```
Main: Before call function
x=3
My_func: x=2
Main: After call function x=2
```

ตัวอย่างนี้แสดงถึงตัวแปรภายนอกชื่อ x ซึ่งเป็นที่รู้จักในทุก ๆ ฟังก์ชันของโปรแกรม (ฟังก์ชันไม่ต้องประกาศก็สามารถเรียกตัวแปร x ใช้ได้) ดังนั้น ดังนั้นเมื่อฟังก์ชัน my_func ถูกเรียกใช้ซึ่งค่าในตัวแปร x ในฟังก์ชัน my_func ถูกเปลี่ยนค่าให้เป็น 2 ดังนั้นค่า x ที่แสดงในฟัง main จึงเปลี่ยนเป็น 2 เหมือนกัน ดังผลการทำงานของโปรแกรม