

```

1  #include <stdio.h>
2  int main()
3  {
4      int a,b,c;
5      a=b=c=0;
6      int i=0,j=0;
7      do{
8          printf("\nOld A : %d -> Enter A : ",a);
9          i= scanf("%d",&a);
10         j= printf("value A : %d \n",a);
11         printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
12     }while(i!=0);
13     printf("\nOld B : %d -> Enter B : ",b);
14     i= scanf("%d",&b);
15     j= printf("value B : %d \n",b);
16     printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
17 }while(i!=0);
18 printf("\nOld C : %d -> Enter C : ",c);
19 i= scanf("%d",&c);
20 j= printf("value C : %d \n",c);
21 printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
22 fflush(stdin);
23 }while(a!=0);
24 printf("End Program\n");
25 return 0;
26 }

```

Build messages:

```

== Build file: "no target" in "no project" (compiler: unknown) ==
== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ==

```

การกำหนดตัวแปรเป็นจำนวนเต็ม (INT)

กรณีที่ 1 มีตัวอักษรปน

เมื่อใส่ INPUT ที่มีตัวอักษรเข้าไป จะทำให้ตัวโปรแกรมเกิดข้อผิดพลาดทำให้ไม่สามารถรับค่า
ต่อได้

Old A : 0 -> Enter A = 3ab value A : 3 Value of scanf : 1 Value of printf : 13	Old A : 35 -> Enter A : g3y value A : 35 Value of scanf : 0 Value of printf : 14
Old B : 0 -> Enter B = : value B : 0 Value of scanf : 0 Value of printf : 15	Old B : 0 -> Enter B : : value B : 0 Value of scanf : 0 Value of printf : 15
Old C : 0-> Enter C = : value A : 0 Value of scanf : 0 Value of printf : 15	Old C : 0 -> Enter C : : value A : 0 Value of scanf : 0 Value of printf : 15

กรณีที่ 2 ทศนิยมเกิน

เมื่อใส่ INPUT ที่เป็นเลขทศนิยมจะทำให้ โปรแกรมรับค่าตัวเลขที่อยู่ข้างหน้าเครื่องหมาย “ . ” แหะหลังจากเครื่องหมาย “ . ” จะทำให้โปรแกรมเกิดข้อผิดพลาดและไม่สามารถรับค่าต่อได้

<pre>Old A : 35 -> Enter A : 1.2.3.4 value A : 1 Value of scanf : 1 Value of printf : 13 Old B : 0 -> Enter B : : value B : 0 Value of scanf : 0 Value of printf : 15 Old C : 0 -> Enter C : : value A : 0 Value of scanf : 0 Value of printf : 15</pre>	<pre>Old A : 1 -> Enter A : 5.7.8.9 value A : 5 Value of scanf : 1 Value of printf : 13 Old B : 0 -> Enter B : : value B : 0 Value of scanf : 0 Value of printf : 15 Old C : 0 -> Enter C : : value A : 0 Value of scanf : 0 Value of printf : 15</pre>
---	--

กรณีที่ 3 +- ปนกัน

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - จะทำให้โปรแกรมสามารถรันได้เป็นปกติ เพราะ + - เป็นเครื่องหมายที่ใช้ในค่าของจำนวนเต็ม คือ จำนวนเต็มบวก และจำนวนเต็มลบ ซึ่งเมื่อใส่เครื่องหมาย และไม่มีตัวเลขตามหลังจะทำให้ข้ามตัวแปรนั้นไป

<pre>Old A : 2 -> Enter A : + value A : 2 Value of scanf : 0 Value of printf : 13 Old B : 5 -> Enter B : - value B : 5 Value of scanf : 0 Value of printf : 13 Old C : 7 -> Enter C : + value C : 7 Value of scanf : 0 Value of printf : 13</pre>	<pre>Old A : 2 -> Enter A : +-90+ value A : 2 Value of scanf : 0 Value of printf : 13 Old B : 5 -> Enter B : value B : -90 Value of scanf : 1 Value of printf : 15 Old C : 7 -> Enter C : value C : 7 Value of scanf : 0 Value of printf : 13</pre>
--	--

กรณีที่ 4 มีเว้นวรรคหลัง + -

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - มีเครื่องหมายช่องว่าง จะทำให้โปรแกรม จะถือว่า ตัวแรกคือเครื่องหมายซึ่งจากกรณีที่ 3 คือข้ามตัวแปรนี้ไป และตัวหลังช่องว่างจะเป็นค่าที่รับตัวถัดไป

<pre>Old A : 2 -> Enter A : + 69 value A : 2 Value of scanf : 0 Value of printf : 13 Old B : -90 -> Enter B : value B : 69 Value of scanf : 1 Value of printf : 14</pre>	<pre>Old A : 2 -> Enter A : - 18 value A : 2 Value of scanf : 0 Value of printf : 13 Old B : 69 -> Enter B : value B : 18 Value of scanf : 1 Value of printf : 14</pre>
---	--

กรณีที่ 5 เลขยกกำลัง

ไม่สามารถใช้เลขยกกำลังได้ เพราะถือว่า ตัว e เป็นตัวอักษร ซึ่งทำให้โปรแกรมเกิดข้อผิดพลาดไม่สามารถรับค่าต่อได้

<pre>Old A : 1 -> Enter A : -1e-9 value A : -1 Value of scanf : 1 Value of printf : 14 Old B : 18 -> Enter B : value B : 18 Value of scanf : 0 Value of printf : 14 Old C : 3 -> Enter C : value C : 3 Value of scanf : 0 Value of printf : 13</pre>	<pre>Old A : 2 -> Enter A : 1e8 value A : 1 Value of scanf : 1 Value of printf : 13 Old B : 18 -> Enter B : value B : 18 Value of scanf : 0 Value of printf : 14 Old C : 3 -> Enter C : value C : 3 Value of scanf : 0 Value of printf : 13</pre>
---	--

```

1 #include <stdio.h>
2 int main()
3 {
4     float a,b,c;
5     a=b=c=0;
6     int i=0,j=0;
7     do{
8         printf("\nOld A : %.2f -> Enter A : ",a);
9         i= scanf("%f",&a);
10        j= printf("value A : %.2f \n",a);
11        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
12        int printf(const char*, ...);
13        printf("\nOld B : %.2f -> Enter B : ",b);
14        i= scanf("%f",&b);
15        j= printf("value B : %.2f \n",b);
16        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
17
18        printf("\nOld C : %.2f -> Enter C : ",c);
19        i= scanf("%f",&c);
20        j= printf("value C : %.2f \n",c);
21        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
22
23        fflush(stdin);
24    }while(a!=10000);
25
26    printf("End Program\n");
27    return 0;
28 }
29

```

การกำหนดตัวแปรเป็น FLOAT

กรณีที่ 1 มีตัวอักษรปน

เมื่อใส่ INPUT ที่มีตัวอักษรเข้าไป จะทำให้ตัวโปรแกรมเกิดข้อผิดพลาดทำให้ไม่สามารถรับค่า
ต่อได้

<pre> Old A : 10.00 -> Enter A : 4a9 value A : 4.00 Value of scanf : 1 Value of printf : 16 Old B : 20.00 -> Enter B : value B : 20.00 Value of scanf : 0 Value of printf : 17 Old C : 30.00 -> Enter C : value C : 30.00 Value of scanf : 0 Value of printf : 17 </pre>	<pre> Old A : 4.00 -> Enter A : 00i value A : 0.00 Value of scanf : 1 Value of printf : 16 Old B : 20.00 -> Enter B : value B : 20.00 Value of scanf : 0 Value of printf : 17 Old C : 30.00 -> Enter C : value C : 30.00 Value of scanf : 0 Value of printf : 17 </pre>
---	--

กรณีที่ 2 ทศนิยมเกิน

เมื่อใส่ INPUT ที่เป็นเลขทศนิยมจะทำให้ โปรแกรมรับค่าตัวแปรแรก จะรับค่าตามปกติ แต่ตัวที่เกินมาถัดไปจะรับข้อมูลเหมือนอ่านค่าได้ 0.x

```
Old A : 0.00 -> Enter A : 1.2.3.4
value A : 1.20
Value of scanf : 1
Value of printf : 16

Old B : 20.00 -> Enter B : value B : 0.30
Value of scanf : 1
Value of printf : 16

Old C : 30.00 -> Enter C : value C : 0.40
Value of scanf : 1
Value of printf : 16
```

```
Old A : 0.00 -> Enter A : 99.87.44.33
value A : 99.87
Value of scanf : 1
Value of printf : 17

Old B : 0.00 -> Enter B : value B : 0.44
Value of scanf : 1
Value of printf : 16

Old C : 0.00 -> Enter C : value C : 0.33
Value of scanf : 1
Value of printf : 16
```

กรณีที่ 3 +- ปนกัน

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - จะทำให้โปรแกรมสามารถรันได้เป็นปกติ เพราะ + - เป็นเครื่องหมายที่ใช้ในค่าของจำนวนเต็ม คือ จำนวนเต็มบวก และจำนวนเต็มลบ ซึ่งเมื่อใส่เครื่องหมายและไม่มีตัวเลขตามหลังจะทำให้ข้ามตัวแปรนั้นไป

```
Old A : 5.00 -> Enter A : +
value A : 5.00
Value of scanf : 0
Value of printf : 16

Old B : 6.00 -> Enter B : -
value B : 6.00
Value of scanf : 0
Value of printf : 16

Old C : 12.00 -> Enter C : +
value C : 12.00
Value of scanf : 0
Value of printf : 17
```

```
Old A : 5.00 -> Enter A : --85-
value A : 5.00
Value of scanf : 0
Value of printf : 16

Old B : 6.00 -> Enter B : value B : -85.00
Value of scanf : 1
Value of printf : 18

Old C : 12.00 -> Enter C : value C : 12.00
Value of scanf : 0
```

กรณีที่ 4 มีเว้นวรรคหลัง + -

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - มีเครื่องหมายช่องว่าง จะทำให้โปรแกรม จะถือว่า ตัวแรกคือเครื่องหมายซึ่งจากกรณีที่ 3 คือข้ามตัวแปรนี้ไป และตัวหลังช่องว่างจะเป็นค่าที่รับตัวถัดไป

```
Old A : 5.00 -> Enter A : - 15
value A : 5.00
Value of scanf : 0
Value of printf : 16
```

```
Old B : -85.00 -> Enter B : value B : 15.00
Value of scanf : 1
Value of printf : 17
```

```
Old A : 5.00 -> Enter A : + 20
value A : 5.00
Value of scanf : 0
Value of printf : 16
```

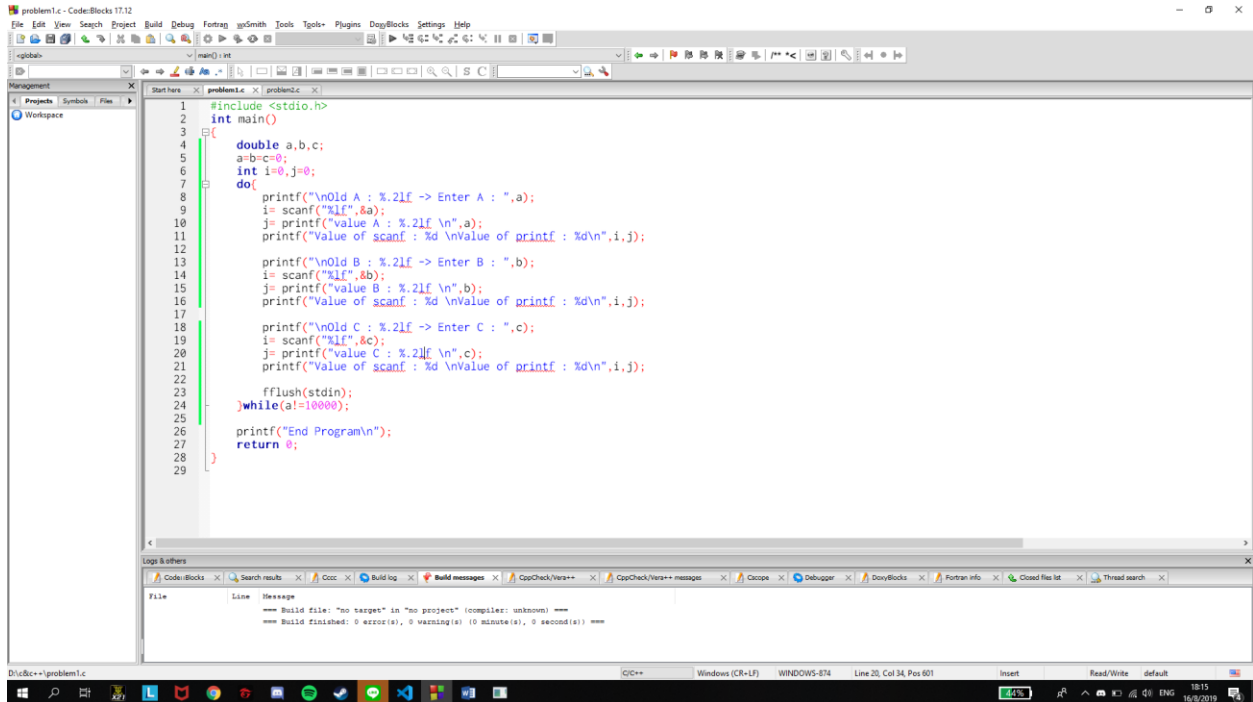
```
Old B : 15.00 -> Enter B : value B : 20.00
Value of scanf : 1
Value of printf : 17
```

กรณีที่ 5 เลขยกกำลัง

สามารถใช้เลขยกกำลังได้

```
Old A : 5.00 -> Enter A : 5e8
value A : 500000000.00
Value of scanf : 1
Value of printf : 24
```

```
Old B : -0.00 -> Enter B : -9e-2
value B : -0.09
Value of scanf : 1
Value of printf : 17
```



```

1 #include <stdio.h>
2 int main()
3 {
4     double a,b,c;
5     a=b=c=0;
6     int i=0,j=0;
7     do{
8         printf("\nOld A : %.2lf -> Enter A : ",a);
9         i= scanf("%lf",&a);
10        j= printf("value A : %.2lf \n",a);
11        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
12
13        printf("\nOld B : %.2lf -> Enter B : ",b);
14        i= scanf("%lf",&b);
15        j= printf("value B : %.2lf \n",b);
16        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
17
18        printf("\nOld C : %.2lf -> Enter C : ",c);
19        i= scanf("%lf",&c);
20        j= printf("value C : %.2lf \n",c);
21        printf("Value of scanf : %d \nValue of printf : %d\n",i,j);
22
23        fflush(stdin);
24    }while(a!=10000);
25    printf("End Program\n");
26    return 0;
27 }
28
29

```

การกำหนดตัวแปรเป็น DOUBLE

กรณีที่ 1 มีตัวอักษรปน

เมื่อใส่ INPUT ที่มีตัวอักษรเข้าไป จะทำให้ตัวโปรแกรมเกิดข้อผิดพลาดทำให้ไม่สามารถรับค่า
ต่อได้

<pre> Old A : 0.00 -> Enter A : 3x2 value A : 3.00 Value of scanf : 1 Value of printf : 16 Old B : 0.00 -> Enter B : value B : 0.00 Value of scanf : 0 Value of printf : 16 Old C : 0.00 -> Enter C : value C : 0.00 Value of scanf : 0 Value of printf : 16 </pre>	<pre> Old A : 9.00 -> Enter A : qr5 value A : 9.00 Value of scanf : 0 Value of printf : 16 Old B : 0.00 -> Enter B : value B : 0.00 Value of scanf : 0 Value of printf : 16 Old C : 0.00 -> Enter C : value C : 0.00 Value of scanf : 0 Value of printf : 16 </pre>
--	--

กรณีที่ 2 ทศนิยมเกิน

เมื่อใส่ INPUT ที่เป็นเลขทศนิยมจะทำให้ โปรแกรมรับค่าตัวแปรแรก จะรับค่าตามปกติ แต่ตัวที่เกินมาถัดไปจะรับข้อมูลเหมือนอ่านค่าได้ 0.x

<pre>Old A : 9.00 -> Enter A : 155.12313.123123.44512 value A : 155.12 Value of scanf : 1 Value of printf : 18 Old B : 0.00 -> Enter B : value B : 0.12 Value of scanf : 1 Value of printf : 16 Old C : 0.00 -> Enter C : value C : 0.45 Value of scanf : 1 Value of printf : 16</pre>	<pre>Old A : 155.12 -> Enter A : 14.22.31241.123 value A : 14.22 Value of scanf : 1 Value of printf : 17 Old B : 0.12 -> Enter B : value B : 0.31 Value of scanf : 1 Value of printf : 16 Old C : 0.45 -> Enter C : value C : 0.12 Value of scanf : 1 Value of printf : 16</pre>
---	---

กรณีที่ 3 + - ปนกัน

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - จะทำให้โปรแกรมสามารถรับได้เป็นปกติ เพราะ + - เป็นเครื่องหมายที่ใช้ในค่าของจำนวนเต็ม คือ จำนวนเต็มบวก และจำนวนเต็มลบ ซึ่งเมื่อใส่เครื่องหมาย และไม่มีตัวเลขตามหลังจะทำให้ข้ามตัวแปรนั้นไป

<pre>Old A : 14.22 -> Enter A : --- value A : 14.22 Value of scanf : 0 Value of printf : 17 Old B : 0.31 -> Enter B : value B : 0.31 Value of scanf : 0 Value of printf : 16 Old C : 0.12 -> Enter C : value C : 0.12 Value of scanf : 0 Value of printf : 16</pre>	<pre>Old A : 14.22 -> Enter A : --+20 value A : 14.22 Value of scanf : 0 Value of printf : 17 Old B : 0.31 -> Enter B : value B : 0.31 Value of scanf : 0 Value of printf : 16 Old C : 0.12 -> Enter C : value C : 20.00 Value of scanf : 1 Value of printf : 17</pre>
--	---

กรณีที่ 4 มีเว้นวรรคหลัง + -

เมื่อใส่ INPUT ที่มีเครื่องหมาย + - มีเครื่องหมายช่องว่าง จะทำให้โปรแกรม จะถือว่า ตัวแรกคือเครื่องหมายซึ่งจากกรณีที่ 3 คือข้ามตัวแปรนี้ไป และตัวหลังช่องว่างจะเป็นค่าที่รับตัวถัดไป

```
Old A : 14.22 -> Enter A : - 200
value A : 14.22
Value of scanf : 0
Value of printf : 17
```

```
Old B : 0.31 -> Enter B : value B : 200.00
Value of scanf : 1
Value of printf : 18
```

```
Old A : 14.22 -> Enter A : + 200
value A : 14.22
value of scanf : 0
value of printf : 17
```

```
Old B : 200.00 -> Enter B : value B : 200.00
value of scanf : 1
value of printf : 18
```

กรณีที่ 5 เลขยกกำลัง

สามารถใช้เลขยกกำลังได้

```
Old C : 20.00 -> Enter C : 9e3
value C : 9000.00
Value of scanf : 1
Value of printf : 19
```

```
Old A : 14.22 -> Enter A : 8.3e2
value A : 830.00
Value of scanf : 1
Value of printf : 18
```

ความเข้าใจใน Assignment นี้

จากการทำ Assignment ในครั้งนี้ ทำให้ ได้ทราบว่า การใช้ตัวในการรับค่า นั้นต้องมีลักษณะการใส่ข้อมูลที่ตรงกับลักษณะของตัวแปรที่กำหนด การใช้คำสั่ง scanf หรือ printf นั้น ถือว่าเป็นฟังก์ชันจะได้ค่าที่ return ออกมาเป็นจำนวนเต็ม ซึ่ง scanf นั้นจะ return ค่าออกมาเป็นจำนวนเต็มของจำนวนตัวแปรที่รับค่ามาจากคำสั่ง scanf ที่ถูกต้องกับ Format Specifiers หากไม่ถูกต้อง จะไม่นับ และ printf จะ return ค่าออกมาเป็นจำนวนตัวอักษร ที่อยู่ในคำสั่ง/ฟังก์ชัน printf นั้น ซึ่งรวมนับรวม Enter และ Space bar ด้วย

INT รับค่าด้วย %d

```
#include <stdio.h>
int main()
{
    int A;

    scanf("%d",&A);

    printf("Use \\%d :%d\\n",A);
    printf("Use \\%f :%f\\n",A);
    printf("Use \\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :1
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d สามารถแสดงค่าได้

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะแสดงผลไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %lf ไม่สามารถแสดงค่าได้ เพราะแสดงผลไม่ตรง Format Specifiers ของตัวแปร

INT รับค่าด้วย %f

```
#include <stdio.h>
int main()
{
    int A;

    scanf("%f",&A);

    printf("Use \\%d :%d\\n",A);
    printf("Use \\%f :%f\\n",A);
    printf("Use \\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :1065353216
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะแสดงผลไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะแสดงผลไม่ตรง Format Specifiers ของตัวแปร

INT รับค่าด้วย %lf

```
#include <stdio.h>
int main()
{
    int A;

    scanf("%lf",&A);

    printf("Use \\\%d :%d\\n",A);
    printf("Use \\\%f :%f\\n",A);
    printf("Use \\\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :0
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรง Format Specifiers ของตัวแปร
และแสดงผลไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรง Format Specifiers ของตัวแปร
และแสดงผลไม่ตรง Format Specifiers ของตัวแปร

FLOAT รับค่าด้วย %d

```
#include <stdio.h>
int main()
{
    float A;

    scanf("%d",&A);

    printf("Use \\\%d :%d\\n",A);
    printf("Use \\\%f :%f\\n",A);
    printf("Use \\\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :0
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะรับค่าและแสดงผลไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะตอนรับ รับค่าไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะตอนรับ รับค่าไม่ตรง Format Specifiers ของตัวแปร

FLOAT รับค่าด้วย %f

```
#include <stdio.h>
int main()
{
    float A;

    scanf("%f",&A);

    printf("Use \\\%d :%d\\n",A);
    printf("Use \\\%f :%f\\n",A);
    printf("Use \\\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :0
Use %f :1.000000
Use %lf :1.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะแสดงผลไม่ตรง Format Specifiers ของตัวแปร

แสดงผลด้วย %f สามารถแสดงค่าได้

แสดงผลด้วย %lf สามารถแสดงค่าได้ เพราะการแสดงผลด้วย %f กับ %lf ถือว่าเป็นการแสดงผลข้อมูลที่มี Format Specifiers เหมือนกัน

FLOAT รับค่าด้วย %lf

```
#include <stdio.h>
int main()
{
    float A;

    scanf("%lf", &A);

    printf("Use %%d :%d\n", A);
    printf("Use %%f :%f\n", A);
    printf("Use %%lf :%lf\n", A);

    return 0;
}
```

```
1
Use %d :0
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะรับค่าและแสดงผลไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะรับค่าไม่ตรงกับ Format Specifiers ของตัวแปร

DOUBLE รับค่าด้วย %d

```
#include <stdio.h>
int main()
{
    double A;

    scanf("%d",&A);

    printf("Use \\%d :%d\\n",A);
    printf("Use \\%f :%f\\n",A);
    printf("Use \\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :1
Use %f :0.000000
Use %lf :0.000000
```

แสดงผลด้วย %d สามารถแสดงค่าได้

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะตอนรับไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะตอนรับไม่ตรงกับ Format Specifiers ของตัวแปร

DOUBLE รับค่าด้วย %f

```

#include <stdio.h>
int main()
{
    double A;

    scanf("%f",&A);

    printf("Use \\%d :%d\\n",A);
    printf("Use \\%f :%f\\n",A);
    printf("Use \\%lf :%lf\\n",A);

    return 0;
}

```

```

1
Use %d :1065353216
Use %f :0.000000
Use %lf :0.000000

```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะตอนรับและแสดงผลไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %f แสดงค่าไม่ถูกต้อง เพราะตอนรับไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %lf แสดงค่าไม่ถูกต้อง เพราะตอนรับไม่ตรงกับ Format Specifiers ของตัวแปร

DOUBLE รับค่าด้วย %lf

```
#include <stdio.h>
int main()
{
    double A;

    scanf("%lf",&A);

    printf("Use \\%d :%d\\n",A);
    printf("Use \\%f :%f\\n",A);
    printf("Use \\%lf :%lf\\n",A);

    return 0;
}
```

```
1
Use %d :0
Use %f :1.000000
Use %lf :1.000000
```

แสดงผลด้วย %d แสดงค่าไม่ถูกต้อง เพราะตอนรับและแสดงผลไม่ตรงกับ Format Specifiers ของตัวแปร

แสดงผลด้วย %f สามารถแสดงค่าได้ ได้ เพราะการแสดงผลด้วย %f กับ %lf ถือว่าเป็นการแสดงผลข้อมูลที่มี Format Specifiers เหมือนกัน

แสดงผลด้วย %lf สามารถแสดงค่าได้

ปัญหาใน Assignment นี้

ในตอนแรกที่ได้รับคำสั่งนั้น ไม่เข้าใจว่าให้ทำอะไร เนื่องจากฟังแล้วตีความโจทย์ไม่แตก

ความเข้าใจใน Assignment นี้

ได้ทราบถึงลักษณะการรับและการแสดงค่าของข้อมูลที่สามารถใช้ได้กับตัวแปรนั้นๆ หากใช้ได้ไม่ถูกต้อง จะทำให้ค่าที่แสดงออกหรือรับมานั้นไม่ถูกต้องและ Format ซึ่งในการรับค่าโดยใช้คำสั่ง scanf นั้น จำเป็นต้องรับด้วยลักษณะการรับข้อมูลของตัวแปรโดยเฉพาะ คือ int ใช้ %d , float ใช้ %f , double ใช้ %lf หากการ scanf นั้นไม่ตรงกับลักษณะของตัวแปรจะทำให้ค่าที่ได้รับมาไม่ถูกต้อง หรือไม่ได้รับค่า ในส่วนของการแสดงค่าออกมานั้น การใช้คำสั่ง printf ก็จำเป็นต้องมีลักษณะสอดคล้องกับลักษณะของข้อมูลเช่นเดียวกัน แต่ การ printf ของ float และ double สามารถใช้ %f หรือ %lf ก็ได้ เนื่องจากมีลักษณะข้อมูลที่คล้ายกัน ในส่วนของ Format Double ซึ่ง scanf ด้วย %d และ printf ด้วย %d สามารถรับค่าและแสดงค่าเป็นจำนวนเต็มได้นั้น ไม่สามารถหาเหตุผลได้

การประเมินตัวเองใน Assignment นี้

Criterion 1 : 60 คะแนน (Competent Performer)

20	40	60	80	100
----	----	----	----	-----

การใช้เครื่องมือในการสร้างโปรแกรม ความเข้าใจในการใช้คำสั่ง : 80 คะแนน (ทำโจทย์ได้ด้วยตัวเอง มีปัญหาบ้าง ยังไม่มีความมั่นใจที่จะทำโจทย์อื่นที่คล้ายกันได้)

20	40	60	80	100
----	----	----	----	-----