2021
計算機プログラミング演習

第5回

if と演算子(==, !=, <, >, <=, >=)

第5回の目標

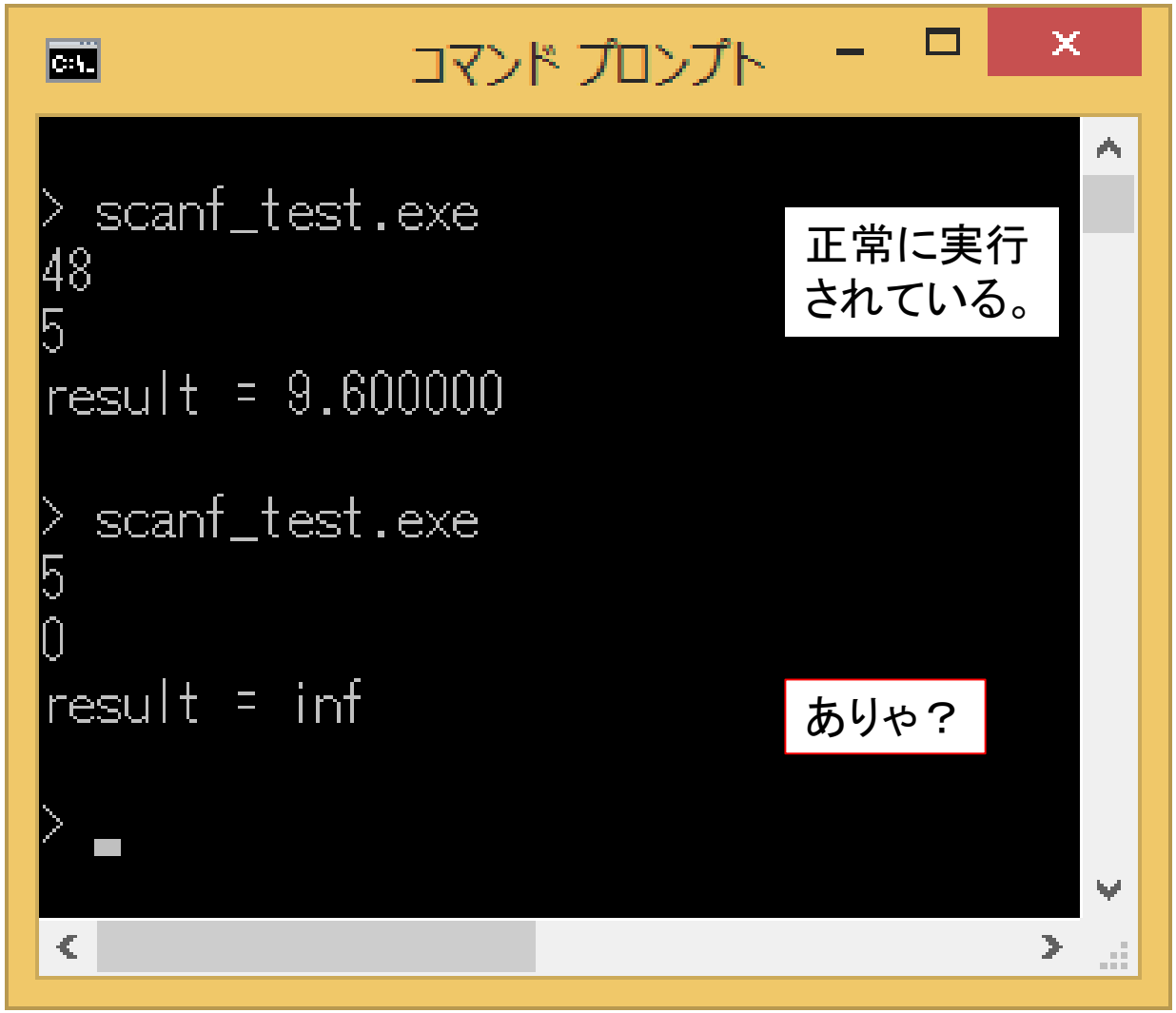
- if による条件判断
  - 条件によって手順や処理を変える
  - else の使い方
  - 条件式とは？
    - 条件式を表示
    - 複数の条件式
  - 一連の処理をブロックとして考える（構造化）

scanf() を使ったプログラム

```
/*     scanf_test.c       */

#define  _CRT_SECURE_NO_WARNINGS 1
#include  <stdio.h>

int  main( void )
{
       double dvalue1, dvalue2, dresult;

       scanf("%lf", &dvalue1 ) ;
       scanf("%lf", &dvalue2 ) ;

       dresult = dvalue1 / dvalue2;
       printf("result = %f¥n", dresult);

       return 0;

}
```

scanf関数でどうしてもエラーが出る場合は scanf_s関数を使用してもよいが，文字列のときは引数の指定の仕方が異なるので注意！

ありゃ？　なんじゃこら？



" inf " に注目！

0（ゼロ）での
割り算を行ったので
無限大（ INFinity）が
出た。

inf は「数ではない」
という数である。

printf() で表示すると
" inf "　と表示されるが、
プログラムは引き続き
実行される。

" inf " が出た dresult
変数を使う演算は
これ以降信用できる
結果が得られない！

if　の使い方

- if による条件判別(1)

  条件式 に合致すれば（真）　*処理* を行う。

  条件式 に合致しなければ（偽）　*処理* を行わない。

  処理 は複数行にわたっても構わない。

```
if （ 条件式 ）
{
        処理 ;
}
```

if の使い方

```
/*      scanf_test.c          */

#define   _CRT_SECURE_NO_WARNINGS 1
#include   <stdio.h>

int   main( void )
{
        double dvalue1, dvalue2, dresult;

        scanf("%lf", &dvalue1 ) ;
        scanf("%lf", &dvalue2 ) ;

        if ( dvalue2 != 0 )
        {
                dresult = dvalue1 / dvalue2;
                printf("result = %f¥n", dresult);
        }

        return 0;

}
```

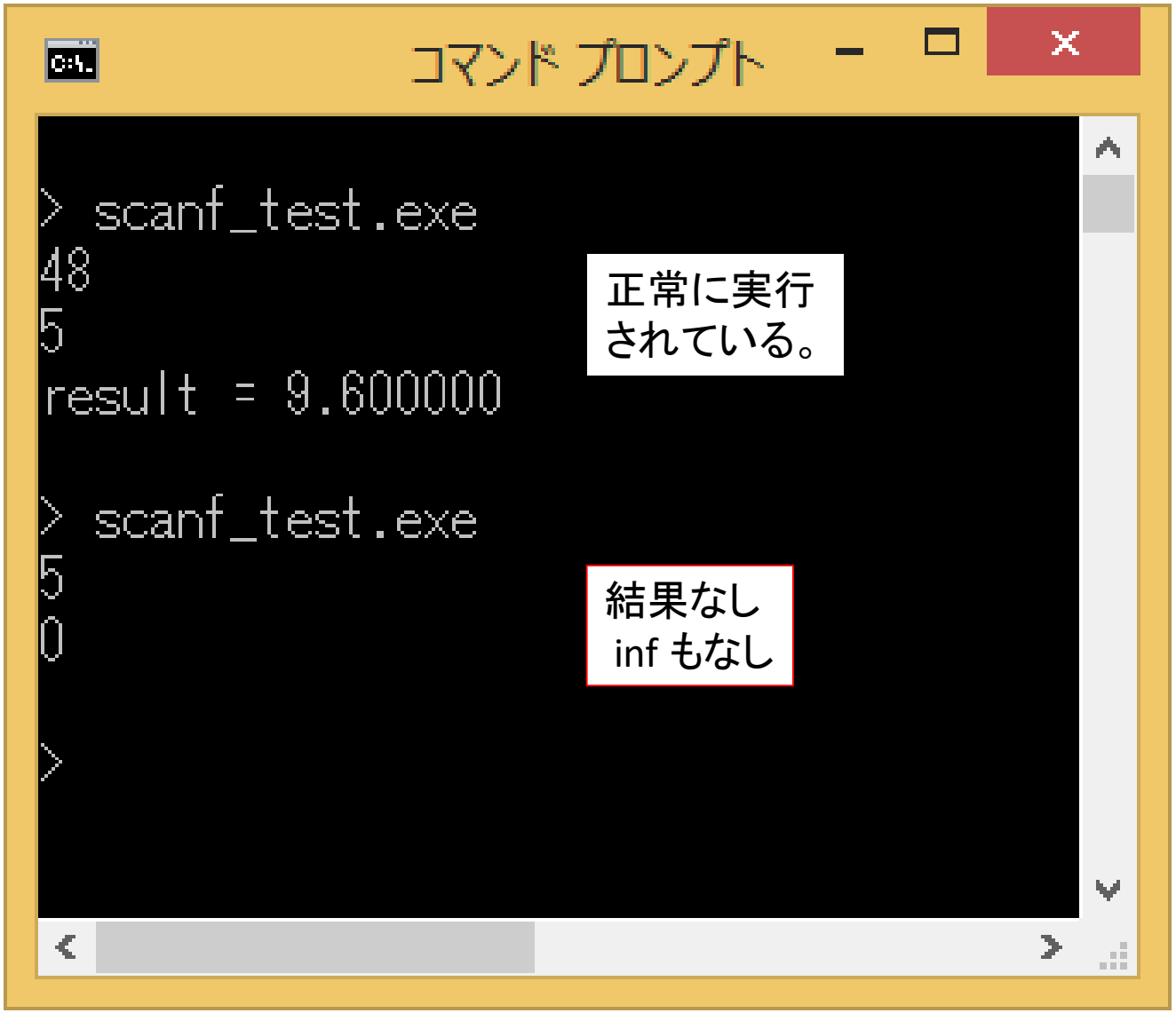
条件が一致した場合の処理が2行以上のとき

！は否定（でない）を意味する

dvalue2 が 0 ではない

中括弧 { } で括る

inf を回避したが…


コマンド プロンプト
> scanf_test.exe
48
5
result = 9.600000
> scanf_test.exe
5
0
>
正常に実行されている。
結果なし
inf もなし

if　の使い方

- if による条件判別(2)
  条件式 に合致すれば(真であれば) *処理1* を行う。
  条件式 に合わなければ(偽であれば) *処理2* を行う。

```
if ( 条件 )
{

        処理1 ;

} else {

        処理2 ;

}
```

if の使い方

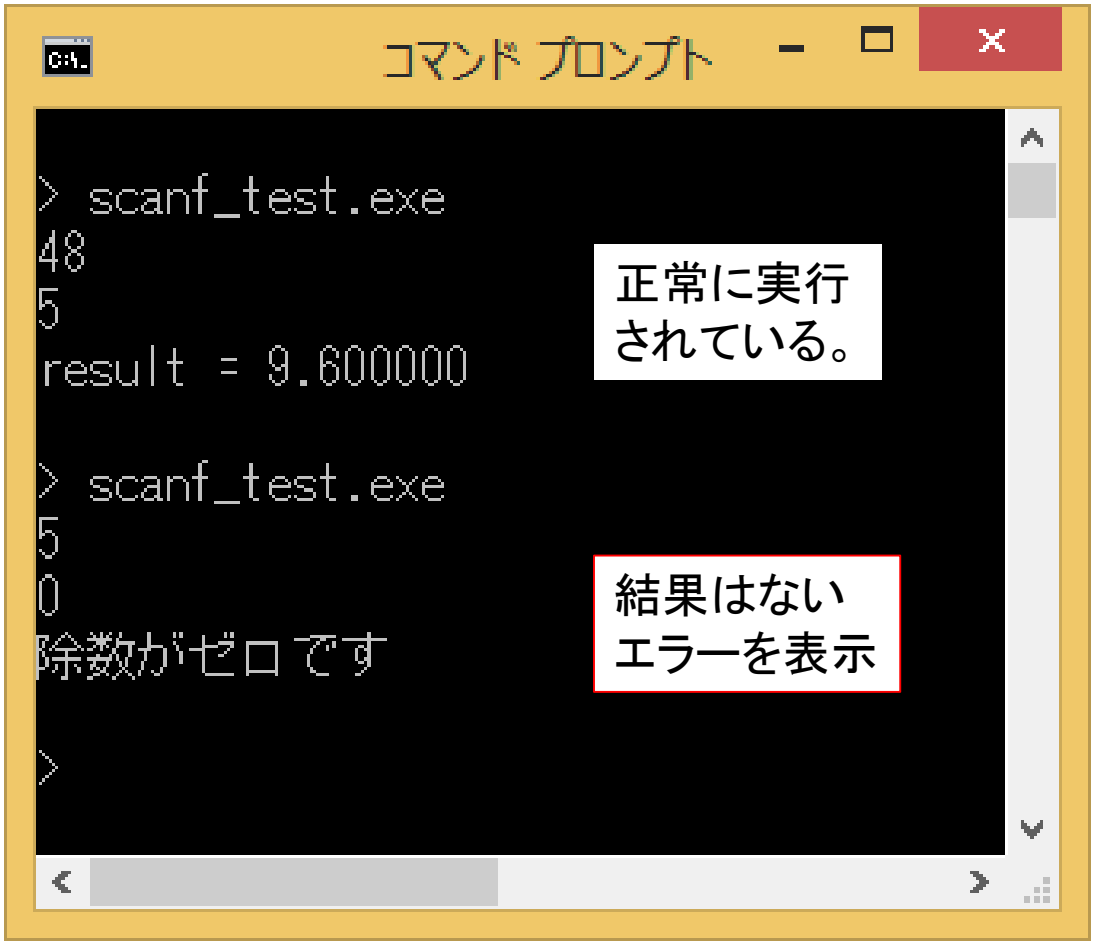
ゼロの時にも
メッセージを出したい。

```
        if ( dvalue2 == 0 )
        {
                printf("除数がゼロです¥n" ) ;
        } else {
                dresult = dvalue1 / dvalue2;
                printf("result = %f¥n", dresult);
        }
        return 0;
}
```

ゼロの時(真)

ゼロでない時(偽)

inf を回避し、異常を通知する



if の使い方

- よく間違う点
  - 条件内で等号(==) と代入(=) を間違う

```
if ( dvalue2 = 0 ) {
```

誤り

```
if ( dvalue2 == 0 ) {
```

正しい

  - 間違っていてもエラーが出ないので要注意！！
  - エラーは出ないが、正しい結果も出ない！！

間違いやすい例（1）

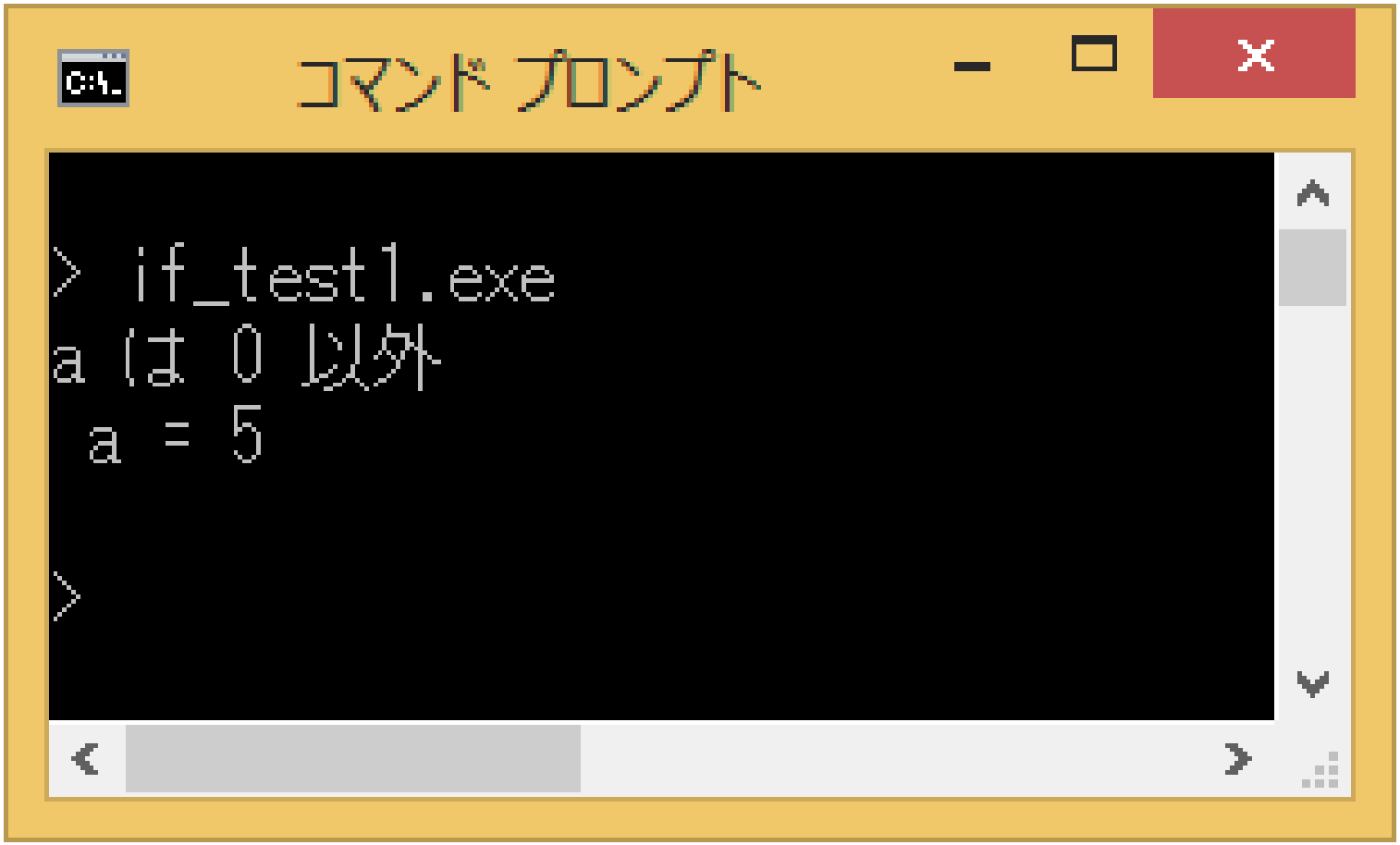
（エラーにならないから要注意）

ビルドしても
正常終了

```
/*    if_test1.c     */

#include  <stdio.h>

int  main(void)
{
        int  a = 0;

        if ( a = 5 )
        {
                printf("a は 0 以外 ¥n");
        }

        printf(" a = %d¥n", a);

        return 0;

}
```

正しくは a == 5 であるが
誤って a = 5 としてしまった

ところが実行してみると



- if の中の「a は 0以外」は表示されている！
- if の後では a の値は 5 に変更されている！

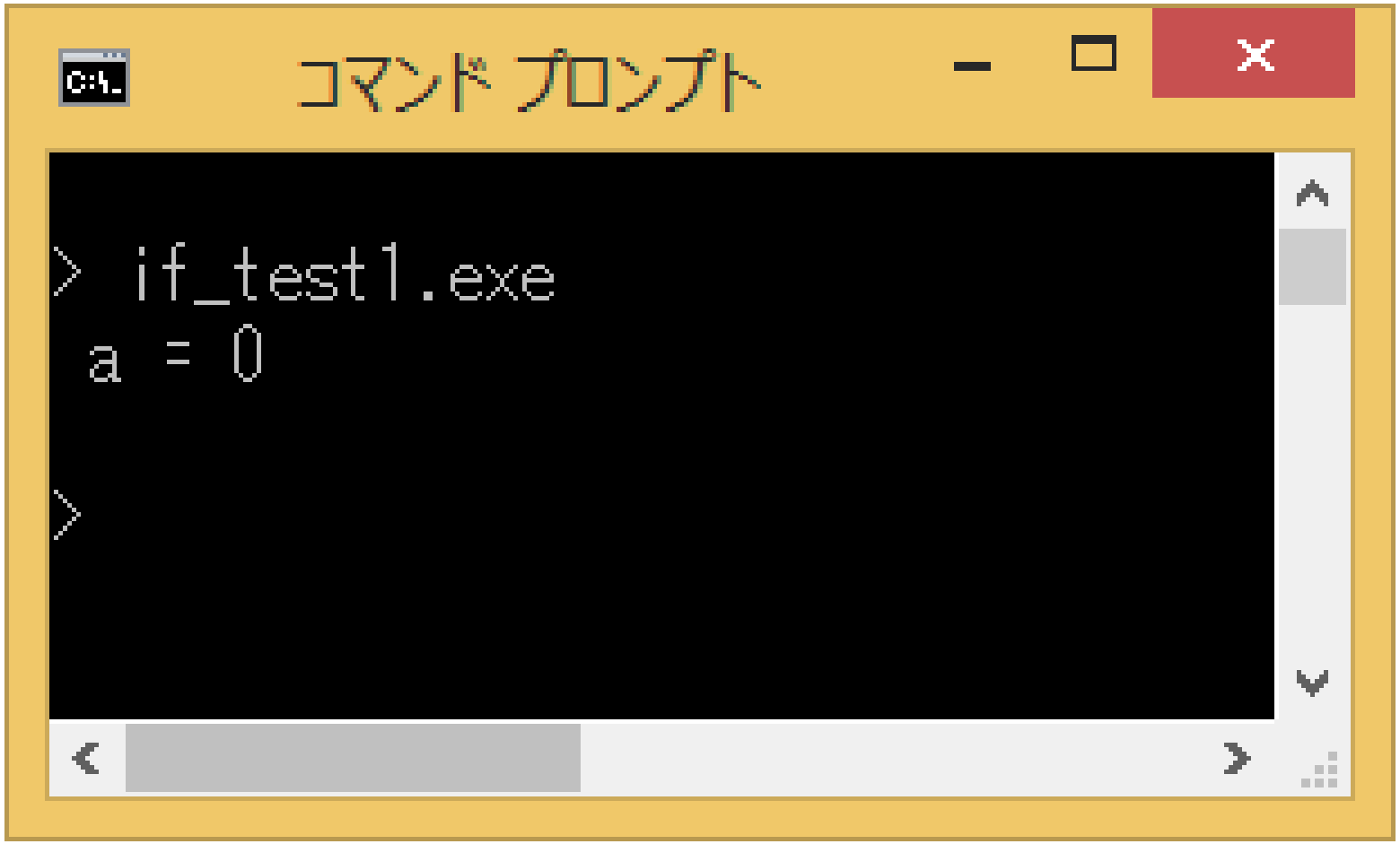
間違いやすい例(2)

```
/*    if_test1.c    */

#include  <stdio.h>

int   main(void)
{
        int   a = 1;

        if ( a = 0 )
        {
                printf("a はゼロ¥n");
        }

        printf(" a = %d¥n", a);

        return 0;

}
```

正しくは a == 0 であるが
誤って a = 0 としてしまった

ビルドしても
正常終了

実行してみると結果が異なる！



- if の中の「a はゼロ」は表示されていない???
- しかし、if の後では a の値は 0 になっている！

条件式について

条件式が真 (True) の場合

```
if ( 条件式 )
{
    処理 ;
}
```

カッコ内の処理を実行

条件式が偽 (False) の場合

```
if ( 条件式 )
{
    処理 ;
}
```

カッコ内の処理を実行しない

```
if ( k )
{
    処理 ;
}
```

if (k) 変数 k (整数) が

- 0 以外の場合 ⇒ 真
- 0 の時 ⇒ 偽

間違いやすい例(2)

```
/*    if_test1.c    */

#include  <stdio.h>

int   main(void)
{
        int   a = 1;

        if ( a = 0 )
        {
                printf("a はゼロ¥n");
        }

        printf(" a = %d¥n", a);

        return 0;

}
```

If (a = 0)だと条件式は偽

カッコ内の処理を実行しない

ビルドしても
正常終了

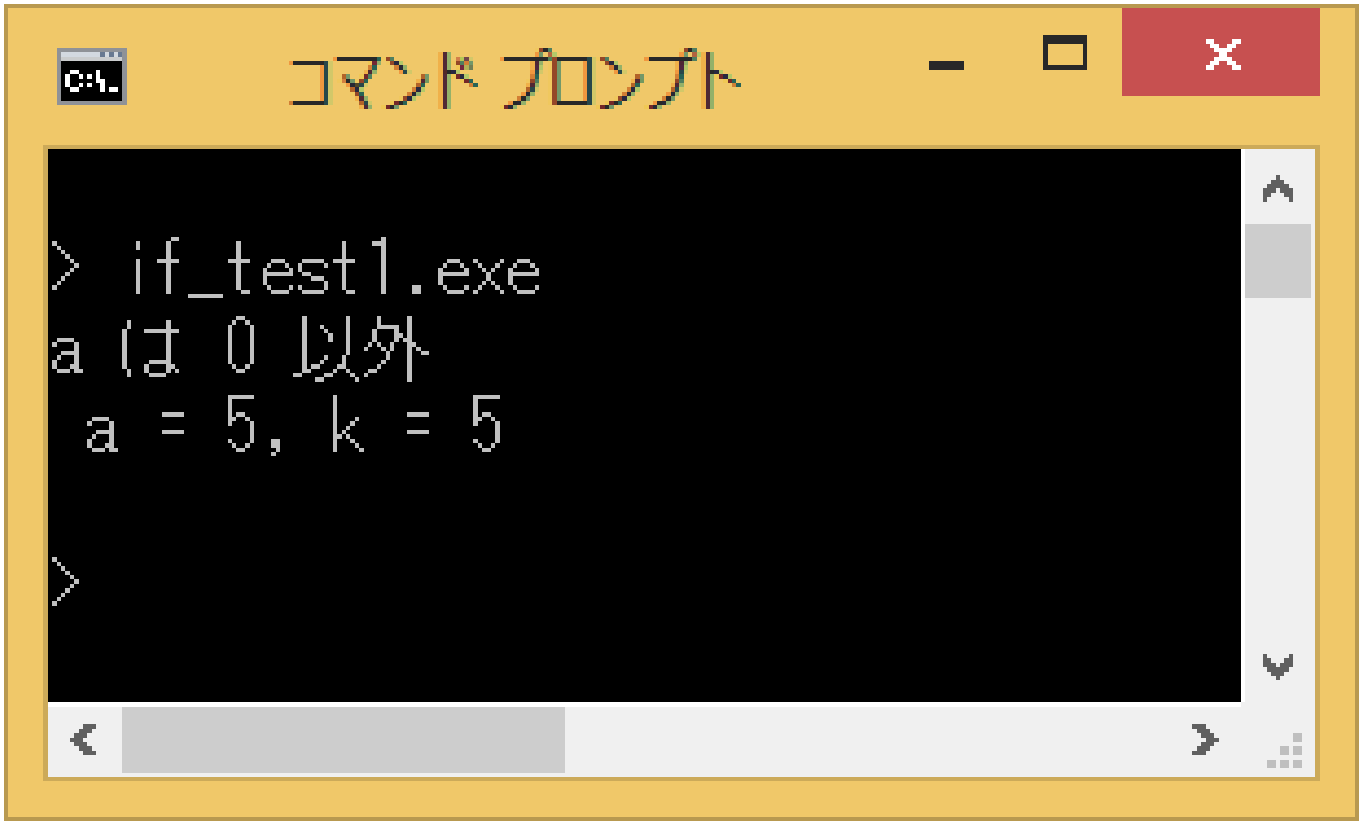
条件式について

if (k) 変数 k (整数)が

- 0 以外の場合 ⇒ 真
- 0 の時 ⇒ 偽

```
/*     if_test1.c    */
#include  <stdio.h>

int  main(void)
{
       int   a = 0, k;

       k = ( a = 5 ) ;

       if ( k )
       {
             printf("a は 0 以外 ¥n");
       }
       printf(" a = %d, k = %d¥n", a, k);
       return 0;
}
```

代入文も戻り値を持っている



- k = (a = 5) ;

  a = 5 (代入)を実行したのち、その値(5) が k に代入される。

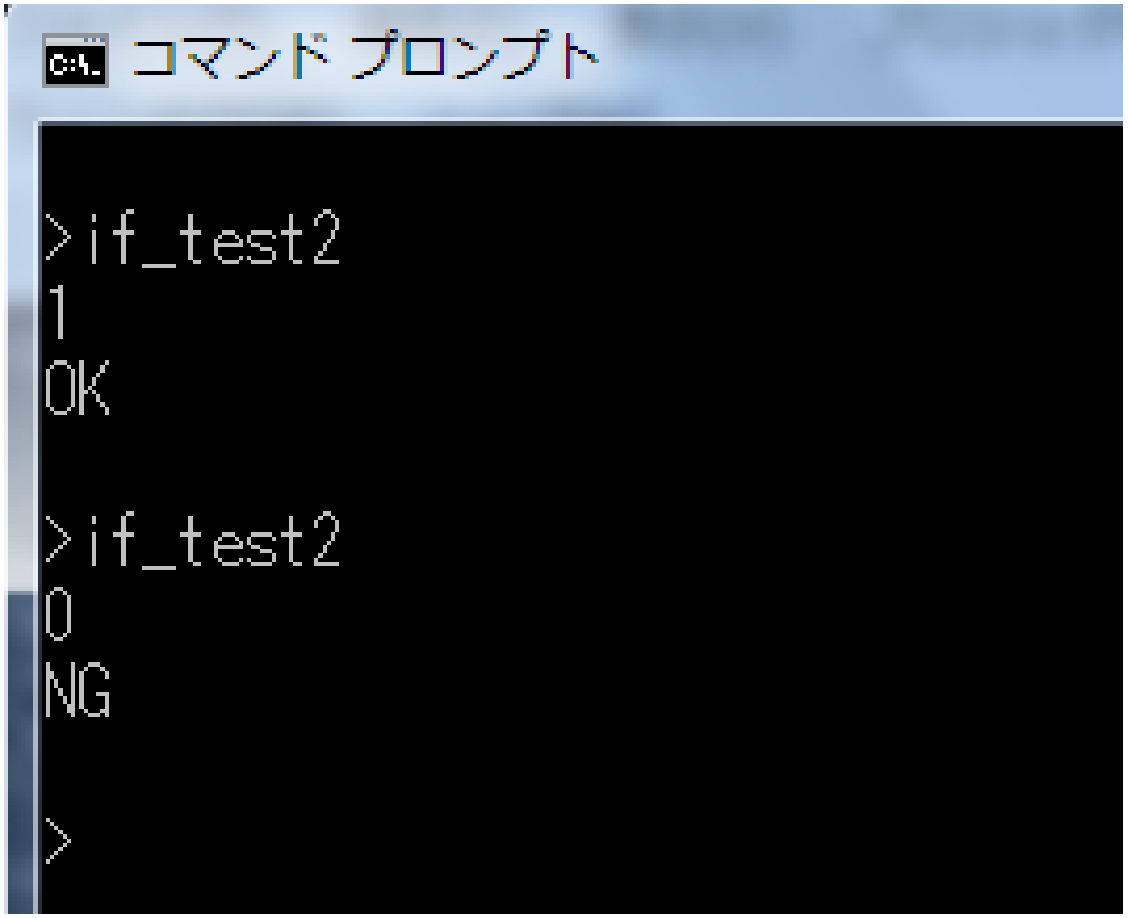
条件式について

if (k) 変数 k (整数)が

- 0 以外の場合 ⇒ 真
- 0 の時 ⇒ 偽

```
/*      if_test2.c    */
#define  _CRT_SECURE_NO_WARNINGS 1
#include  <stdio.h>

int   main(void)
{
        int   k;

        scanf("%d", &k);

        if  (  k  )
       {
                printf("OK¥n");
        } else {
                printf("NG¥n");
        }

        return 0;
}
```

条件式について



- 0 以外の数を k に入力しても NG にならない

条件式について

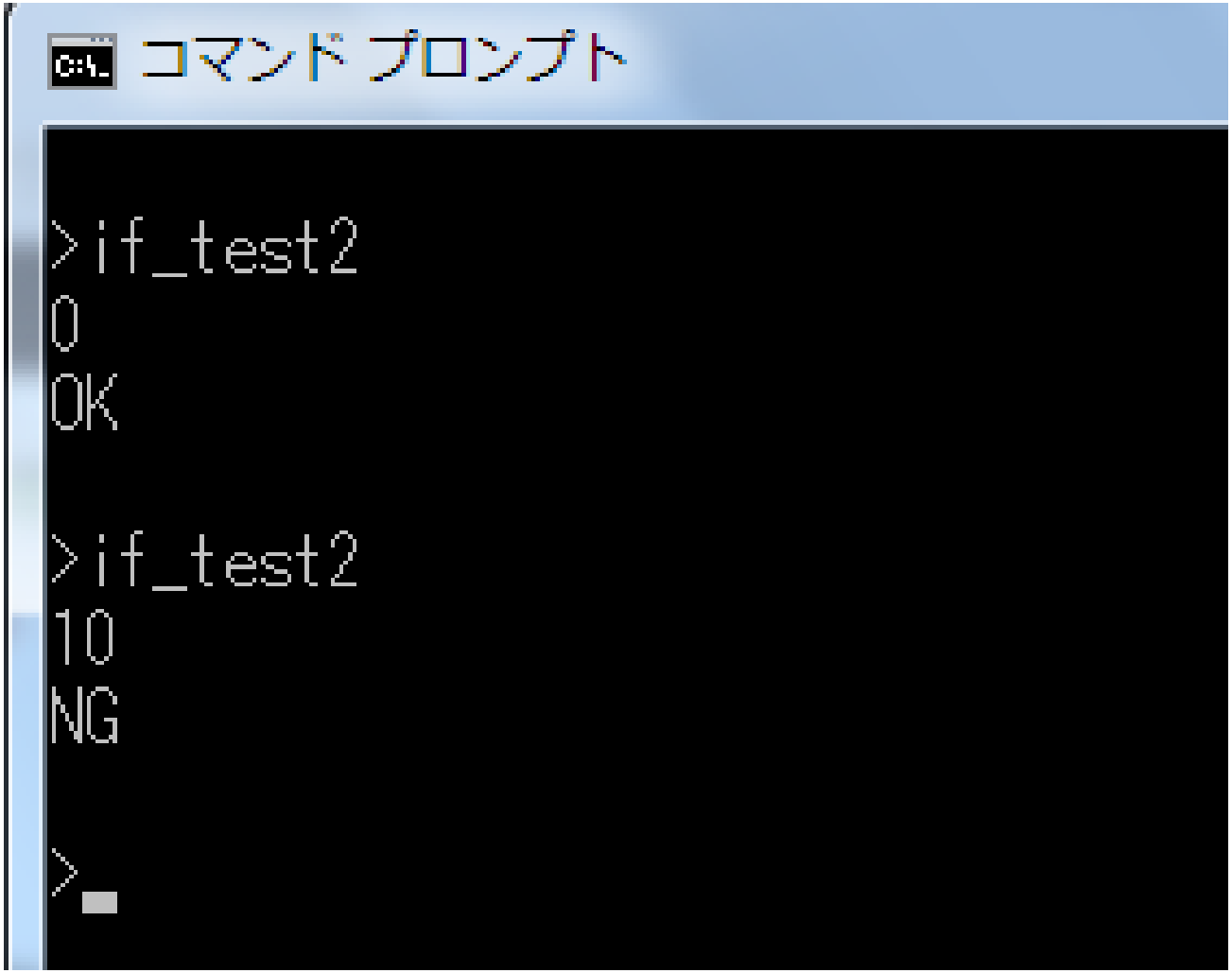
if (k) 変数 k (整数)が
- 0 以外の場合 ⇒ 真
- 0 の時 ⇒ 偽

if (!k) 変数 k (整数)が
- 0 の時 ⇒ 真
- 0 以外の場合 ⇒ 偽

！は否定(でない)を意味する

```
/*     if_test2.c     */
#define  _CRT_SECURE_NO_WARNINGS 1
#include  <stdio.h>

int  main(void)
{
        int  k;

        scanf("%d", &k);
        if ( !k )
        {
                printf("OK¥n");
        } else {
                printf("NG¥n");
        }
        return 0;
}
```

条件式について



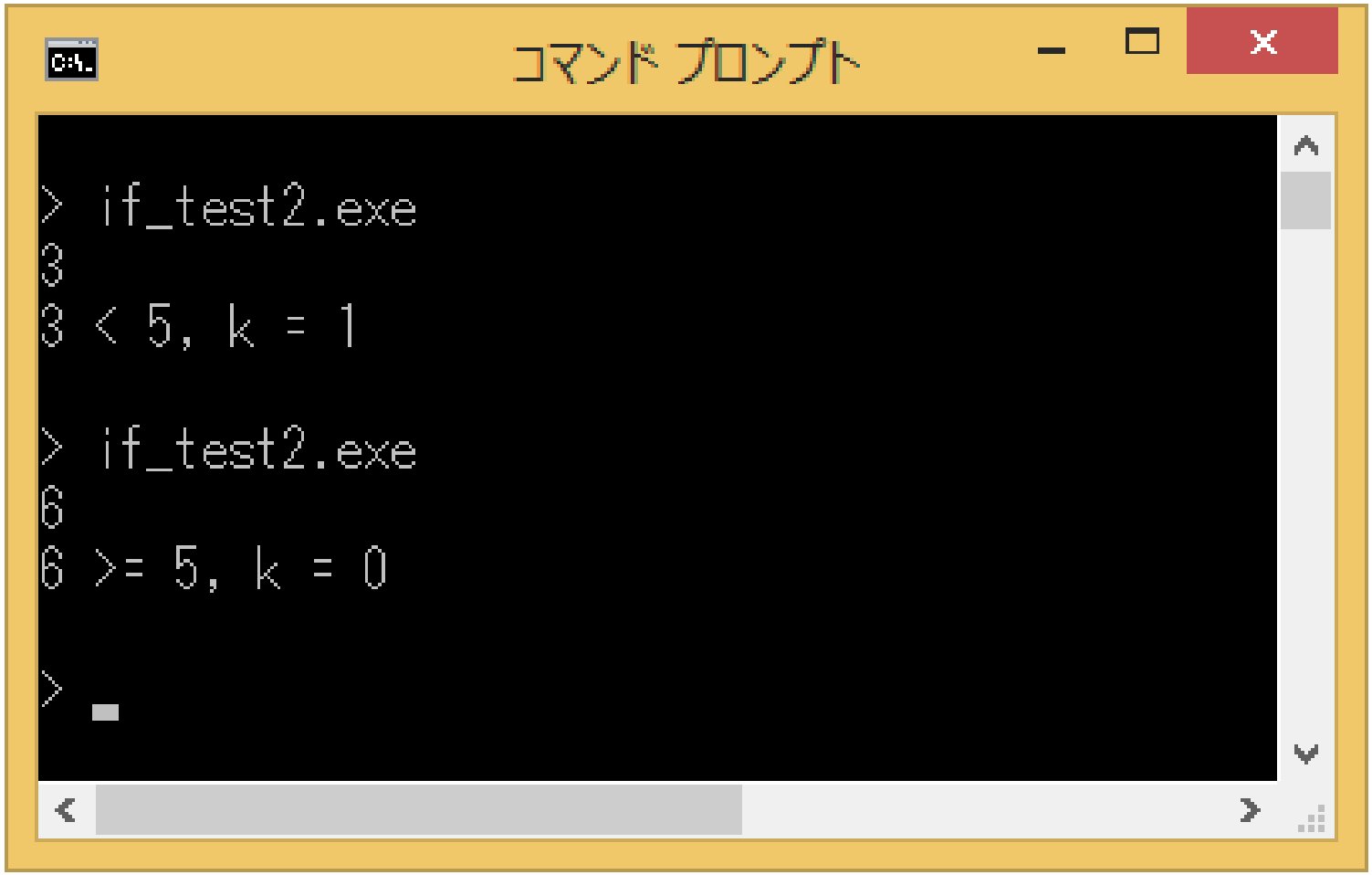
- 0 をｋに入力したときだけ OK になる

条件式について

- 条件式を演算に使う

```
/*      if_test2.c      */
#define  _CRT_SECURE_NO_WARNINGS 1
#include  <stdio.h>

int  main(void)
{
        int   k, m;

        scanf("%d", &m);
        k = ( m < 5 ) ;

        if ( k )
        {
                printf("%d < 5, k = %d¥n", m, k ) ;
        } else {
                printf("%d >= 5, k = %d¥n", m, k );
        }
        return 0;
}
```

条件式について

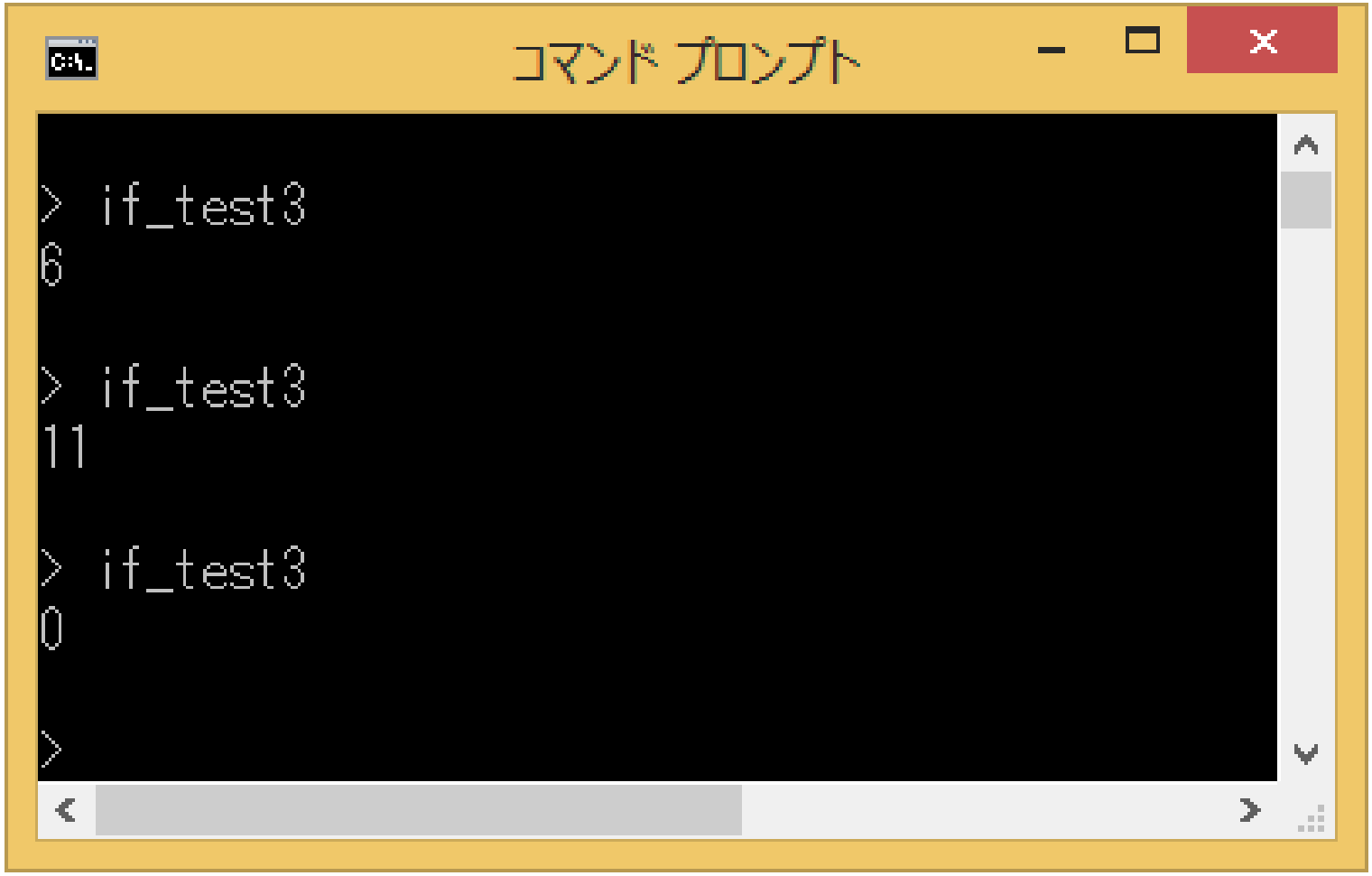


- k = (m < 5) ;　真の時 k = 1,　　偽の時 k = 0

間違えやすい条件

```c
/*    if_test3.c    */

#define  _CRT_SECURE_NO_WARNINGS 1
#include  <stdio.h>

int  main(void) {

      int  a ;

      scanf("%d", &a);

      if ( 10 > a > 5 ) {
            printf(" %d は 6 ~ 9 の数¥n", a);
      }
      return 0;
}
```

間違えやすい条件式



- 常に偽

間違えやすい条件
(このように解釈されている)

```
if ( 10 > a > 5 )
```
→
```
if ( ( 10 > a ) > 5 )
```

```
int  main(void) {

     int  a ;

     scanf("%d", &a);

     if ( ( 10 > a ) > 5 ) {
          printf(" %d は 6 ~ 9 の数¥n", a);
     }
     return 0;
}
```

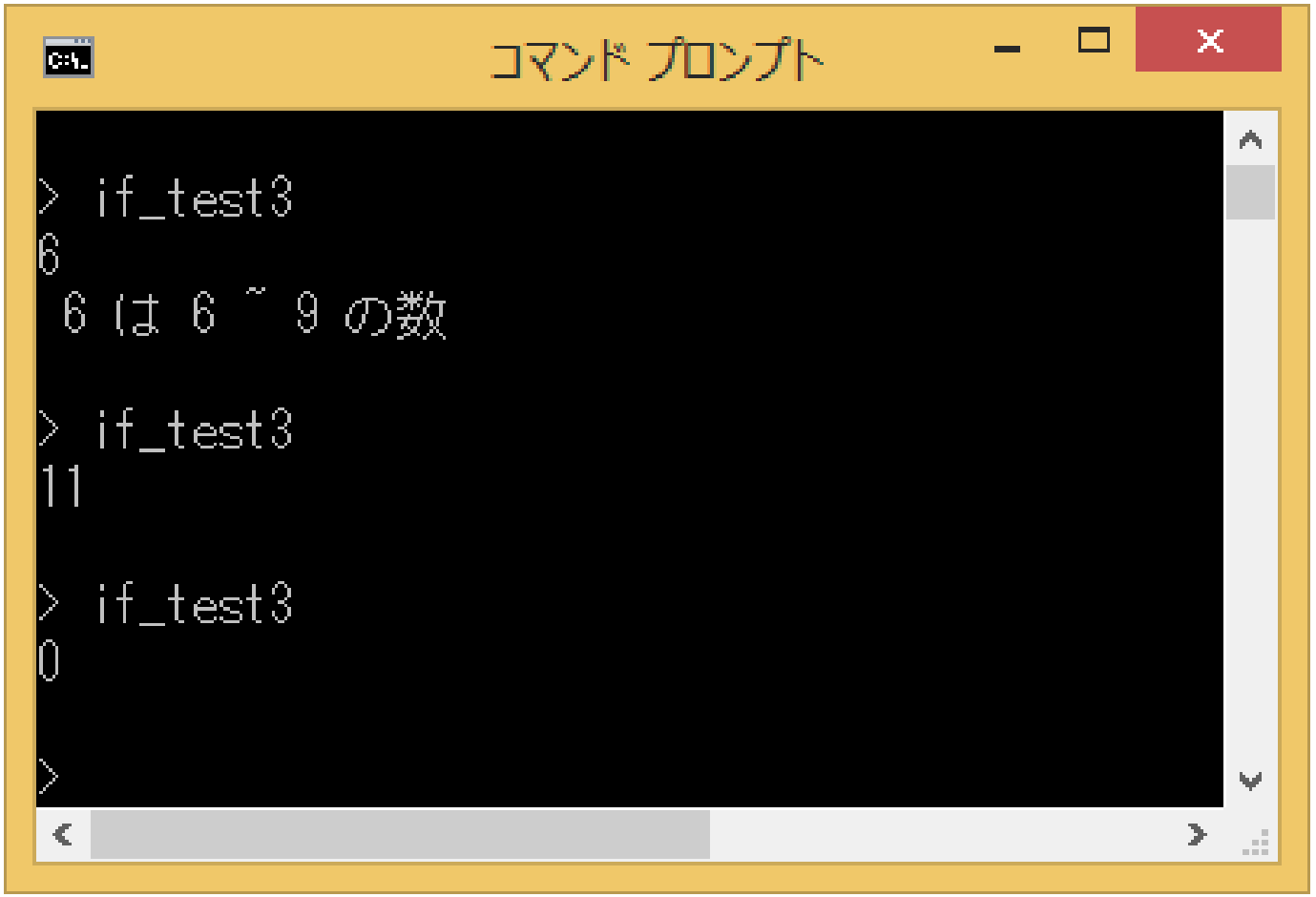
この部分は 0 (偽) か 1 (真) → 0 > 5 , 1 > 5 は どちらも (偽)

&& (論理積) を使って2つの条件に分離する (抜粋)

A && B は　Aかつ（論理積）B
A || B は　Aまたは（論理和）Bという意味

```
scanf("%d", &a);

if ( ( 10 > a ) && ( a > 5 ) )
{

    printf(" %d は 6 ~ 9 の数¥n", a);

}
```

&& (論理積)による2つの条件



- 正しく動作している