○ 正しい対応（浮動小数点数の出力）

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

    p = 3.141593;

    printf("%f¥n", p);

    return 0;
}
```

これは３つの対応が正しい例です．変数pはfloat型（浮動小数点型）で宣言しているので，ここには3.141593という実数を代入することができます．float型の変数を参照するには，%fという変換文字を使用します．

× int型変数に浮動小数点数を入れている

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

    p = 3.141593;

    printf("%d¥n", p);

    return 0;
}
```

変数pをint型で宣言しているので，pには整数値しか代入できないはずなのに，3.141593という実数を代入してしまっています．ただし，この場合はとくにコンパイルエラーとはならず，int型変数pには小数点以下の値(0.141593)を切り捨てた整数値3だけが代入されます．したがって実行結果は，

```
 3
```

として出力されるだけです．

× int型変数に浮動小数点数を入れて%fで出力している

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

    p = 3.141593;

    printf("%f¥n", p);

    return 0;
}
```

int型変数pに実数3.141593を代入してはいけないだけではなく，int型変数を扱う変換指示子は%dでなければならないのに，%fを使用しているところが間違っています．これもコンパイルエラーとはならず，実行型プログラムも作成されますが，実行結果が希望通りになりません．

× float型変数を%dで出力している

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

    p = 3.141593;

    printf("%d¥n", p);

    return 0;
}
```

float型変数pに実数3.141593を代入するところまでは正しいですが，参照のための変換指示子に%dを使用しています．これも実行結果がおかしくなります．

○ 正しい対応（整数と浮動小数点数の代入）

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

    a = 2 * 3;
    b = 2.0 * 3.0;

    printf("%d¥n", a);
    printf("%f¥n", b);

    return 0;
}
```

int型変数aには2*3の結果を代入し，それを%dで参照しているので正しく，float型変数bにも，2.0*3.0というように，実数として表記した数値を使った計算の結果を代入した方が望ましいでしょう．それを%fで参照するのも正解です．

△ float型変数に代入する数値が整数になっている

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

```
    float  a;
    int  b;

    a = 2 * 3;
    b = 2.0 * 3.0;

    printf("%f¥n", a);
    printf("%d¥n", b);

    return 0;
}
```

これは必ずしも間違いとは言えませんが，float型変数aには，2*3ではなく2.0*3.0の結果を代入した方がよいでしょう．また，int型変数bの方に2.0*3.0を代入するのは意味がなく，2*3と書いた方が適切です．

○ 正しい対応

```
#include <stdio.h>
main()
{
    float r, p, ring;

    r = 10.0;
    p = 3.141593;
    ring = r * 2.0 * p;

    printf("%f¥n", r);
    printf("%f¥n", p);
    printf("%f¥n", ring);

    return 0;
}
```

float型変数r,p,ringを使用して計算を行い，それぞれを%fで出力しているので正しい書式です．

△ float型変数ringの計算にint型の数値が混在している

```
#include <stdio.h>
main()
{
    int  r;
    float p, ring;
    r = 10;
    p = 3.141593;
    ring = r * 2.0 * p;

    printf("%d¥n", r);
    printf("%f¥n", p);
    printf("%f¥n", ring);

    return 0;
}
```

これも必ずしも間違っているわけではありませんが，本来float型変数ringの答えを出す計算式には，int型変数rの変数を使わないのが原則です．この場合はコンパイラの方で，int型変数rの値10を自動的に10.0という実数に置き換えて計算します．この場合は，int型変数rを一時的にfloat型変数に変換するために，

```
ring = (double)r * 2.0 * p;
```

と(double)を変数rの前に置いて型変換（キャストという）した方が望ましいでしょう．

以下は，扱うデータが代入する変数の記憶範囲を越えてしまう例です．

× unsigned char型変数を%dで出力している

```
#include <stdio.h>
main()
{
    unsigned char a, b;

    a = 127;
    b = 128;
    printf("%d¥n", a);
    printf("%d¥n", b);

    return 0;
}
```

よく間違えることですが，unsigned型（符号なし）で宣言した変数は%uで参照しなければなりません．%dで参照するとおかしな結果が出ます．なお，char型は－128～127までの数値しか記憶できませんが，これはunsigned char型なので0～255までの値を記憶でき，

```
b = 128;
```

自体は正しい代入です．

○ 正しい対応 unsigned char型変数は%uで出力する

```
#include <stdio.h>
main()
{
    unsigned char a, b;

    a = 127;
    b = 128;
    printf("%u¥n", a);
```

```
    printf("%u¥n", b);

    return 0;
}
```

こちらが正しい参照です．unsigned型（符号なし）で宣言した変数は%uで参照します．

× unsigned short型変数を%dで出力している

```
#include <stdio.h>
main()
{
    unsigned short a, b;

    a = 32767;
    b = 32768;
    printf("%d¥n", a);
    printf("%d¥n", b);

    return 0;
}
```

同じくunsigned型（符号なし）で宣言した変数は%uで参照しなければなりません．

○ 正しい対応（unsigned short型変数は%uで出力する）

```
#include <stdio.h>
main()
{
    unsigned short a, b;

    a = 32767;
    b = 32768;
    printf("%u¥n", a);
    printf("%u¥n", b);

    return 0;
}
```

こちらが正しい参照です．

× unsigned char型変数(b)に入れる数値が範囲を越えている

```
#include <stdio.h>
main()
{
    unsigned char a, b;

    a = 255;
    b = 256;
    printf("%u¥n", a);
    printf("%u¥n", b);

    return 0;
}
```

unsigned char型変数が記憶できるのは0～255までの値だけなので，256は代入できません．この場合も，とくにコンパイルエラーとはなりませんが，実行するとおかしな結果が出ます．

× char型変数(b)に入れる数値が範囲を越えている

```
#include <stdio.h>
main()
{
    char a, b;

    a = 127;
    b = 128;
    printf("%d¥n", a);
    printf("%d¥n", b);

    return 0;
}
```

同じように，char型変数が記憶できるのは－128～127までの値だけなので，128は代入できません．

× unsigned char型変数(a)やunsigned short 型変数(b)に入れる数値が範囲を越えている

```
#include <stdio.h>
main()
{
    unsigned char a;
    unsigned short b;

    a = 255 + 65;
    b = 65535 + 1001;
    printf("%u¥n", a);
    printf("%u¥n", b);

    return 0;
}
```

同じように，unsigned char型変数やunsigned short型変数に代入する値が，それぞれの範囲を越えてしまっています．