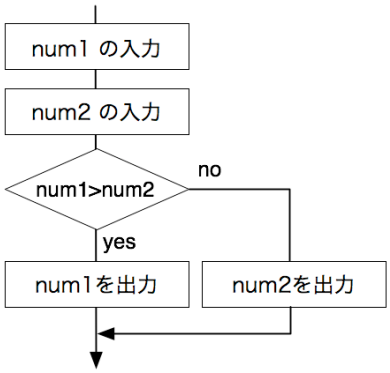
基礎プログラミング演習 II

■ リハビリ演習（つづき）

□ 07：二数のうちの大きい方を出す

入力した二数のうち、より大きな方を出力するプログラムを示します。流れ図と合わせて動作を把握してください。

```
int num1, num2;
scanf("%d", &num1);
scanf("%d", &num2);
printf("num1: %d, num2: %d¥n", num1, num2);
if( num1 > num2 ) {
    printf("answer: %d¥n", num1);
} else {
    printf("answer: %d¥n", num2);
}
```



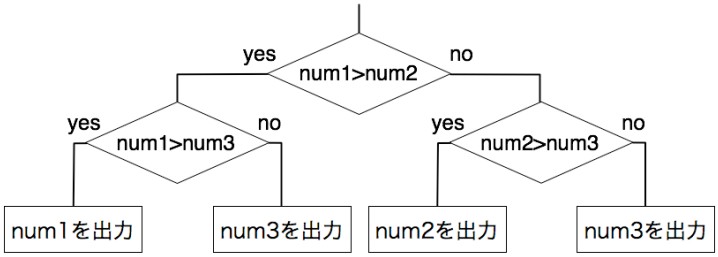

```
$ ./larger01
10
20
num1: 10, num2: 20
answer: 20
$ ./larger01
20
10
num1: 20, num2: 10
answer: 20
$ ./larger01
10 10
num1: 10, num2: 10
answer: 10
$
```

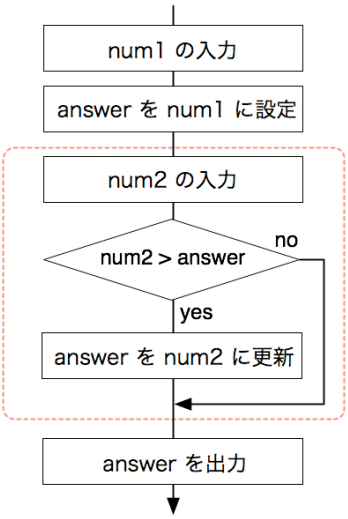
・ 動作確認のやり方

動作確認時の入力値の組み合わせに注目してください。まず 10, 20 と 20, 10 の二通りを入れることで、用意されている条件分岐の両側を通過して、それぞれ正しく機能することを確認しています。
次に 10, 10 を試しているのはそれが「特殊な条件」だからです。このプログラムでは「num1 > num2」あるいは「それ以外」だけで判定しており「num1 == num2」の場合どのようにするか書いていません。実際には同値であった場合は「それ以外」と判定され、num2 が出力されますが、num1 と num2 は同値なので、どちらが出力されてもそれで良いのです。
動作確認はこのように「あり得る状況をすべて想定して網羅的に試験する」ことを覚えてください。

□ 08：三数のうち最も大きいものを出す

次に三つの数のうち最大のものを出す方法を検討しましょう。まず右図のように力ずくで場合分けする方法が思い浮かぶでしょう。（実装としては複数の if 文を入れ子にして組み合わせる）
しかしこれはプログラムを複雑にするだけで、四つ、五つとなったときに応用が利きません。



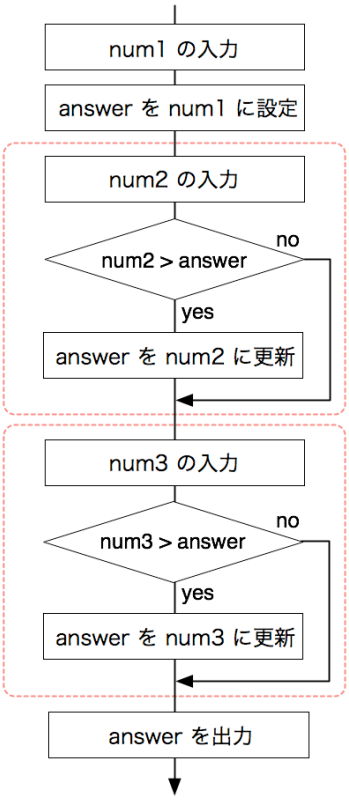


そこで次のように考えます。まず二数の大きい方を出力する方法を左図のように変更します。
方針（アルゴリズム）としては、
1. num1, num2 を直接比較するのではなく、
2. answer という最終結果を保存する変数を用意し、
3. この answer にまず num1 を入れ、
4. 次に answer と num2 と比較し、
5. もし num2 が大きければ answer を更新する

となります。これで最終的に answer に残った値が最大だ、というわけです。

実際のプログラムは右のようになります。
上の 07 の例と同様に動作確認して、正しく三数に対応できていることをチェックしてください。

```
int num1, num2, answer;
scanf("%d", &num1);
printf("num1: %d,¥n", num1);

answer = num1;

scanf("%d", &num2);
printf("num2: %d,¥n", num2);

if( num2 > answer ) {
    answer = num2;
}

printf("answer: %d¥n", answer);
```



そしてよく見ると前頁図、赤の点線枠で囲った部分をもう一度繰り返せば、三つの数に対応できることがわかるでしょうか。左図、赤枠の num2 比較部分を、全く同じ構造で num3 向けに繰り返すだけで三つの数に対応できています。

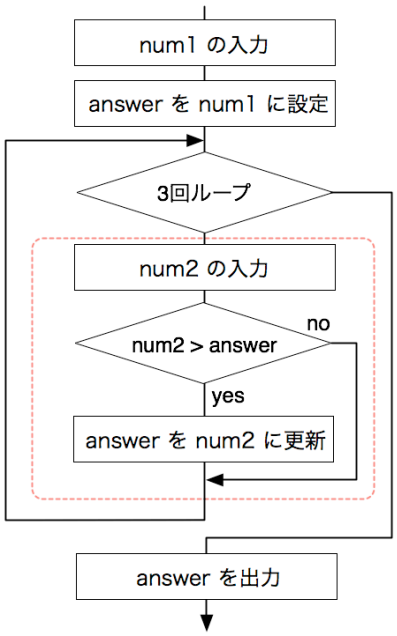
以下にそのプログラムを示します。

```
int num1, num2, num3, answer;
scanf("%d", &num1);
answer = num1;

scanf("%d", &num2);
if( num2 > answer ) {
    answer = num2;
}

scanf("%d", &num3);
if( num3 > answer ) {
    answer = num3;
}

printf("(%d, %d, %d)¥n", num1, num2, num3);
printf("answer: %d¥n", answer);
```

```
$ ./larger03
1 2 3
(1, 2, 3)
answer: 3
$ ./larger03
1 3 2
(1, 3, 2)
answer: 3
$ ./larger03
3 2 1
(3, 2, 1)
answer: 3
$
```

網羅的なテストを行うためにはどのような組み合わせを試す必要があるか、考えて、実際に行ってください。
まず紙にテストケースを列挙して、それを順に入力するのが良いです。

□ 09：繰り返しを用いて四つの数に対応する



赤枠で囲った部分がまるごと二回繰り返されているのを見れば、これが繰り返し処理を用いて書けることがわかるでしょう。つまりループ処理を使って幾つでも数を入力し、そのなかから最大のものを抜き出すことが出来るはずです。
左図はそのように流れ図を書き直したものです。プログラムも例示しますが、どちらもコンパクトにまとまったことがわかるでしょう。

```
int i, num1, num2, answer;

scanf("%d", &num1);
printf("num1: %d,¥n", num1);
answer = num1;

for( i=0; i<3; i++) {
    scanf("%d", &num2);
    if( num2 > answer ) {
        answer = num2;
    }
}

printf("answer: %d¥n", answer);
```

□ 10：任意個数に対応させる

プログラムを修正して、四つの数ではなく、任意回数入力を繰り返せるようにしてください。
つまり、はじめからループ回数を決めるのではなく（ループは無限に繰り返すものとして）、もしデータとして 0 を入力すればそこでループから抜け出して、その時点での最大値を出力するようにしてください。
・ 無限ループは while(1) { ..... } で実現できます
・ ループからの脱出には break 文が使えます

最初のデータ入力でいきなり 0 が来ることは考慮しなくて構いませんが、少し工夫するとスマートに実現出来ます。挑戦推奨です。