1. 今回の目標

   ABC086_A

   が解けるレベルの文法を学ぶ.

   問題を見てみよう.

ABC086A - Product

時間制限 : 2sec / メモリ制限 : 256MB

配点 : 100 点

問題文

シカの AtCoDeer くんは二つの正整数 a,b を見つけました. a と b の積が偶数か奇数か判定してください.

制約

- $1 \leq a,b \leq 10000$
- a,b は整数

入力

入力は以下の形式で標準入力から与えられる.

```
a b
```

出力

積が奇数なら `Odd` と, 偶数なら `Even` と出力せよ.

(https://beta.atcoder.jp/contests/abc086/tasks/abc086_a より引用)

入出力については scanf,printf を使えばよい.積については a*b で計算できる.しかし問題は,

積が奇数なら `Odd` と, 偶数なら `Even` と出力

の部分である.奇数と偶数で処理を分けなくてはいけない.そこで今回は,処理を分けるための「条件分岐」について説明する.

2. コメント

   本題に入る前に,コメントについて説明しておこう.

```
/* 内容 */
```

という書式で,/*と*/で囲まれた処理を行わせないようにできる.囲まれた内容のことを「コメント」といい,/*と*/で囲むことを「コメントアウトする」という.

コメントは主に「ソースコード中にメモを残す」「(バグ発見などのために)一時的にある処理を行わせないようにする」という二点の目的で利用される.

3. 関係演算子・論理演算子

関係演算子とは,条件が真の時に 1,そうでないときに 0 の結果を出す演算子.

関係演算子	意味
A == B;	A と B は等しい
A != B;	A と B は等しくない
A <= B;	A は B 以下
A < B;	A は B より小さい
A >= B;	A は B 以上
A > B;	A は B より大きい

論理演算子は,数学でいう「かつ」とか「または」とか「否定」とかを表現する演算子.

論理演算子	意味
〇 && ×	〇かつ×
〇 || ×	〇または×
!〇	〇の否定

〇や×には先に述べた関係演算子の式が入る.

以下のようにつなげて書くことができる.優先順位を気にしたいなら()でくくる.

〇 && × && △

〇 && (× || △)

実際に関係演算子・論理演算子の値を見てみよう.以下は,入力した 2 つの整数値 a, b に対して,様々な関係演算子,論理演算子を適用した結果を表示するプログラムである.

Sample2_1.c

```c
#include <stdio.h>


int main(void) {
    int a, b;


    scanf("%d %d", &a, &b);
    printf("a == b                  : %d¥n", a == b);
    printf("a > b                   : %d¥n", a > b);
    printf("a + b <= 10             : %d¥n", a + b <= 10);
    printf("a %% 3 == 0 || b %% 3 == 0 : %d¥n", a % 3 == 0 || b % 3 == 0);
    printf("a %% 4 != 0 && b %% 4 != 0 : %d¥n", a % 4 != 0 && b % 4 != 0);
```

```
    return 0;
}
```

実行結果は以下のようになる.ここでは,12 12,4 6,14 10 の三つの入力例を示した.

```
12 12
a == b                    : 1
a > b                     : 0
a + b <= 10               : 0
a % 3 == 0 || b % 3 == 0  : 1
a % 4 != 0 && b % 4 != 0  : 0
```

```
4 6
a == b                    : 0
a > b                     : 0
a + b <= 10               : 1
a % 3 == 0 || b % 3 == 0  : 1
a % 4 != 0 && b % 4 != 0  : 0
```

```
14 10
a == b                    : 0
a > b                     : 1
a + b <= 10               : 0
a % 3 == 0 || b % 3 == 0  : 0
a % 4 != 0 && b % 4 != 0  : 1
```

a==b は「a は b と等しい」

a > b は「a は b より大きい」

a + b <= 10 は「(a+b)は 10 以下である」

a%3==0 || b%3==0 は「a と b の少なくとも一方は 3 の倍数」

a%4 != 0 || b%4 != 0 は「a と b は両方とも 4 の倍数でない」

であることを表す.真のとき 1,偽のとき 0 が表示されているのが分かる.

4. if 文

条件によって処理を分けるには if 文または if-else 文を使う.書式は以下の通りである.

```
if (条件 1) 中身 1
```

```
if (条件 1) 中身 1
else 中身 E
```

条件 1 が 0 以外なら,中身 1 を処理
条件 1 が 0 なら,中身 E を処理する
という単純な文である.もし「条件を満たさなかった場合の処理」を行う必要がないなら,else を書く必要はない.
if 文と else 文を組み合わせて,次の文を書くことがよくある.

```
if (条件 1) 中身 1
else if (条件 2) 中身 2
else if (条件 3) 中身 3
…
else (条件 E) 中身 E
```

条件 1 が 0 以外なら中身 1 を処理.
条件 1 が 0 なら,条件 2 を調べ,それが 0 以外なら中身 2 を処理.
条件 2 が 0 なら,条件 3 を調べ…以下同様.
もしすべての条件が 0 なら中身 E を処理.
という流れで条件分岐を行う.

中身の部分についてだが,原則一つの処理しか書くことができない.
しかし中身の部分を{}でくくると,{}内をひとまとまりの処理とみなすことができるため,複数の処理が書ける.

実際に 0 や 1 を if 文の条件内に書くことはまずない.3 で述べた関係演算子・論理演算子を条件式の中に書くことが多い.
以下は,入力した整数値 a,b に対して,a が b より大きい,小さい,等しいと表示するプログラムである.

Sample2_2.c

```c
#include <stdio.h>

int main(void) {
    int a, b;

    scanf("%d %d", &a, &b);
    if (a == b) {
        printf("%d is equal to %d.¥n", a, b);
    } else if (a > b) {
        printf("%d is bigger than %d.¥n", a, b);
    } else {
```

```
        printf("%d is smaller than %d.¥n", a, b);
    }

    return 0;
}
```

実行結果は以下のようになる.2 2,24 32,43 22 の入力例を示している.

```
2 2
2 is equal to 2.
```

```
24 32
24 is smaller than 32.
```

```
43 22
43 is bigger than 22.
```

まず,次の文に注目しよう.

```
if (a == b) {
  printf("%d is equal to %d.¥n", a, b);
}
```

二つの入力値が 2 2 だとしよう.すると a == b は真(すなわち 1)となるから,

```
if (1) {
  printf("%d is equal to %d.¥n", a, b);
}
```

と読み替えられる.if 文の()内が 0 以外になったため,直後の{ }内の処理が行われる.

二つの入力値が 24 32 だとしよう.すると a == b は偽(すなわち 0)となるから,

```
if (0) {
  printf("%d is equal to %d.¥n", a, b);
}
```

と読み替えられる.if 文の()内が 0 になったため,直後の{ }はスキップされる.次の

```
else if (a > b) {
  printf("%d is bigger than %d.¥n", a, b);
}
```

を見る.24 < 32 のため,a > b も偽(すなわち 0)となる.よって,

```
else if (0) {
  printf("%d is bigger than %d.¥n", a, b);
}
```

に読み替えられ,if(0)直後の { }はスキップされる.そして最後の

```
  else {
    printf("%d is smaller than %d.¥n", a, b);
  }
```

の { }内の処理が行われる.

もう一つ例を示そう.入力された正整数が 7 の倍数かを判定するプログラムである.

```
#include <stdio.h>

int main(void) {
    int a;

    scanf("%d", &a);
    if (a % 7 == 0) {
        printf("7 divides %d.¥n", a);
    } else {
        printf("7 doesn't devide %d.¥n", a);
    }

    return 0;
}
```

実行結果は以下のようになる.49, 64 の入力例を示している.

```
49
7 divides 49.
```

```
64
7 doesn't devide 64.
```

7 の倍数かどうかを調べるには,7 で割った余りが 0 かどうかを調べればよい.よって,if 文の()内には a % 7 == 0 を書いている.

5. switch 文

if 文ほど使う機会がないかもしれないが,一応ここで swtich 文についても説明しておこう.switch 文は,複数の条件分岐を行いたい場合に用いられる.分岐させたいものが 3 つある場合について,以下に書き方の例を示す.

```
switch (式) {
  case 定数 1:
    中身(複数可)
```

```
    break;
  case 定数 2:
    中身(複数可)
    break;
  case 定数 3:
    中身(複数可)
    break;
  default:
    中身(複数可)
}
```

switch 直後の()内の値に応じて,それぞれの case へと分岐する.

式が定数 1 と等しいならば,case 定数 1:の**中身(複数可)**を処理する.

式が定数 2 と等しいならば,case 定数 2:の**中身(複数可)**を処理する.

式が定数 3 と等しいならば,case 定数 3:の**中身(複数可)**を処理する.

式が定数 1 でも定数 2 でも定数 3 でもないならば,default:の**中身(複数可)**を処理する.

ここで,

break;

とは,switch の{ }内から抜けるための文である(厳密には switch だけに用いられる文ではないのだが,詳しくは繰り返し文のところで学ぶ).case の**中身(複数可)**の後に,この文を書くようにしょう.これがないと,例えば case 定数 1: だけを処理したかったのに,それ以降の case 定数 2,case 定数 3 も処理されてしまうので注意が必要である.

さて,switch 文を利用する際の注意点を挙げておこう.

・定数の部分に変数を書くことはできない

・『値が等しい』という比較しかできない

・配列や文字列の比較はできない

・整数値(または文字)同士の比較しかできない

よって,変数同士を比較したかったり,大小関係を調べたかったり,実数値同士を比較したかったりする場合は,if 文を用いる.文字は内部で整数値として扱われているので,switch 文で比較可能.文字列の比較については,C 言語では if 文でもできないので,何か別の手を考えなくてはいけない.それはまた別の機会に述べる.

例を示そう.以下は,入力された正整数に対し,a を 3 で割って 0 余るなら「X」,1 余るなら「Y」,2 余るなら「Z」を表示するプログラムである.

```c
#include <stdio.h>

int main(void) {
    int a;

    scanf("%d", &a);
    switch (a % 3) {
        case 0:
            printf("X¥n");
            break;
        case 1:
            printf("Y¥n");
            break;
        case 2:
            printf("Z¥n");
            break;
        default:
            printf("N¥n");
    }

    return 0;
}
```

実行結果は以下のようになる.10, 98, 24 の入力例を示している.

```
10
Y
```

```
98
Z
```

```
24
X
```

「すべての整数は余りが 0,1,2 なのだから,default 内の処理は行われることはないのでは？」と思うかもしれない.確かにその通りだ.しかし,念のため「0 でも 1 でも 2 でもなかった場合の処理」を書いている.もし default の処理が行われたら,プログラム内部で何か良くないことが行われている.default はこのように,バグ発見のために用いられることもある.

6. 演習

[if 文]

ABC086_A

ABC049_A

ABC081_A

[switch 文]

AtCoder Programming Guide for Beginners Ex6 『電卓を作ろう 』

[早く終わった人向け]

ABC065_A