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 ≤ *a*,*b* ≤ 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.

1. 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.

1. 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 |
| #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.

1. 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 |
| #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.

1. 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.

|  |
| --- |
| #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.

1. 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