5. 制御構造

前章までのプログラムは, 基本的に実行文を上から順に実行していく単純な構造をしていた。実際のプログラムでは, 条件によって実行する文を変更したり, 同じ文を何回も実行したりと, 必ずしも上から順に実行するわけではない。このような, プログラムの流れを決定するものを, プログラムの制御構造と呼ぶ。本章では，制御構造に関係して以下を学習する。

●基本となる制御構造
●順次実行
●分岐実行（if 文や switch 文）
●繰り返し実行（while 文や for 文）

5．1　基本制御構造

すべてのアルゴリズムは，たった 3 つの制御構造の組み合わせで記述することができる。3 つの制御構造とは，図 5．1 に示すような順次実行，分岐実行，繰り返し実行である。

順次実行とは，これまで登場したプログラム例のように，上から順番に 1 つずつ文を実行していく構造である。

分岐実行とは，ある条件によって複数の文の中のどれかを実行する構造である。もっとも単純な分岐は，2 つの文のどちらかを選択して実行するもので，これを特に 2 分岐と呼び，if 文がその代表である。また，Java では，n 分岐のための switch 文も用意されている。

繰り返し実行とは，ある条件が満足されている間は，同じ文が何度でも実行される構造である。Java では，while 文，for 文が用意されている。

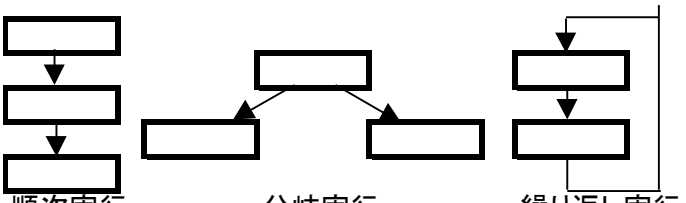

図5．1　3つの基本制御構造

実際のプログラムは，図 5．2 に示すように，これら 3 つの制御構造が組み合わされて使用される。逆に言えば，どんなに複雑なプログラムでもこれら 3 つの制御構造を使えば記述することができる。

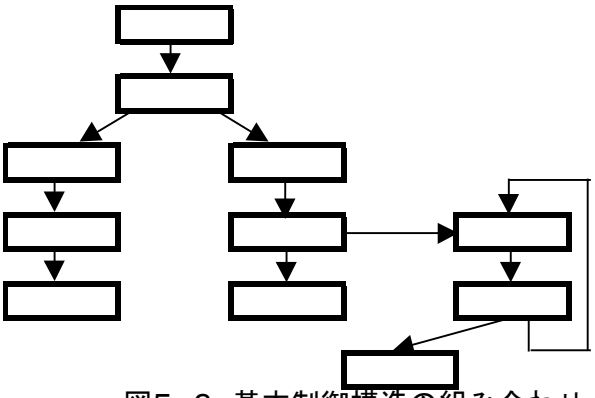

図5．２ 基本制御構造の組み合わせ

5．2　順次実行

Java では，先に出てきた文から順に文 1，文 2 と，順番に実行される。

```
文 1；
文 2；
文 3；
～
文 n；
```

文の切れ目は“；”によって示される。改行，空白，タブなどは文の切れ目とは関係なく，次のように記述しても実行順序は変わらない。

```
文 1；文 2；文 3；～文 n；
```

または，

```
文 1；文 2；
文 3；
文 4；～文 n；
```

などのようにいくらでも書き方はあるが，最終的には読みやすい（理解しやすい）ソースコードになるように気を配るべきである。

Java では，複数の文を中括弧で囲むことにより，囲まれた複数の文を 1 つの文として取り扱うことができる。これをブロックと呼ぶ。例えば，上の例の n 個の文は，次のように記述することにより，1 つの文として取り扱われる。

```
｛　文 1；文 2；文 3；～文 n；｝
```

ブロックは順次実行ではあまり意味を持たないが，分岐実行や繰り返し実行では重要な働きがある。

5．3　if 文

もっとも単純な分岐実行である if 文は，以下のような形式をとる。

if（条件）　文；

1f 文は、条件が真（true）のときは文を実行し，偽（false）の場合は実行しない。プログラムの流れは図 5．3 のようになる。

```
文0;          if文の前の文
if(条件)      条件が真ならば
   文1;         文1を実行
文2;          if文の後の文
```

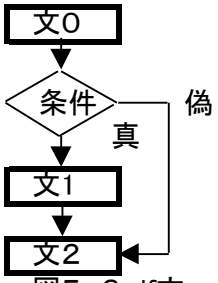

図5．3　if文

条件は関係演算子や論理演算子を使って記述する。例えば，変数 a の値が b の値より小さかったら a<b と表示するプログラムは，以下のように記述する。

```
if（a く b）
        System．out．println（“a く b”）;
```

演習 5．1
整数型変数 a，b に適当な値を代入し，a が b より大きい場合のみ a＞b　と表示するプログラムを作成せよ。

5．4　if－else 文

条件が満足される場合にはある文を，満足されない場合には別の文を実行するという 2 分岐実行は if－else 文により表現される。if－else 文は，以下のような形式をとる。

```
if（条件）　文 1；
else　　文 2；
```

条件が真の場合は文 1 を実行し，偽の場合には文 2 を実行する。プログラムの流れは図 5．4 のようになる。

```
文0;      if文の前の文
if(条件)  条件が真ならば
  文1;      文1を実行
else        条件が真でなければ
  文2;      文2を実行
文3       if文の後の文
```

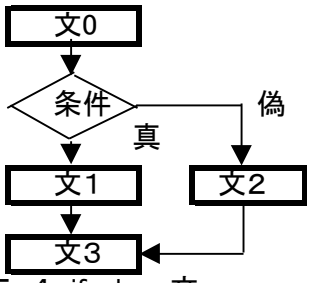

図5．4　if-else　文

演習 5．2
int 型変数 a，b に適当な値を代入し，a と b が等しい値か否かを判断する以下のプログラムを完成させよ。

```
if（a＝＝b）
        System．out．println（“a＝＝b”）;
   else
        System．out．println（“a＞b or a＜b”）;
   System．out．println（” end “）;
```

5．5　ブロックと if 文の入れ子構造

if 文や if－e1se 文を使ってプログラムを書こうとすると，すぐ壁に突き当たってしまう。なぜなら，if や else の後には文を 1 つしか書けないからである。この問題を解決するのが，この章の最初に紹介したブロックである。ブロック化することにより，複数の文を 1 つの文として取り扱うことができるので，図 5．5 のような記述が可能になる。

```
文 0；                    if 文の前の文
if（条件）                条件が真ならば
   {文 1；文 2；文 3；}        文 1、2、3 を実行、
文 4；                    if 文の後の文
```

```
文 0；                    if 文の前の文
if（条件）                条件が真ならば
   {文 1：文 2；文 3；}       文 1，2，3 を実行
else                      条件が真でなければ
   {文 4；文 5；文 6；}       文 4，5，6 を実行
```

文 7　　if 文の後の文

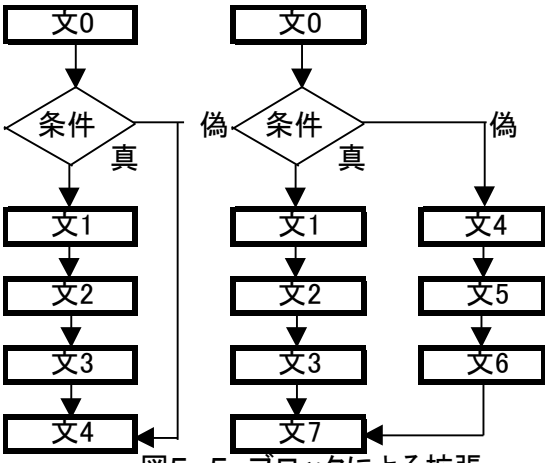

図5．5　ブロックによる拡張

プログラム 5．1 に、ブロックを使ったプログラムの例を示す。このプログラムは，整数 a，b の大小と，差の絶対値を表示する。4，5 行目では，整数を乱数によって生成している。Math.random（）は，0 以上 1 未満の double の値を返す。それに 10 を乗じて int 型にキャストしているので，a や b は 0 から 9 のいずれかの値をとる。変数 c に a と b の差を代入し（8 行目），c の値が 0 より小さい場合は，c の値を絶対値に直してから（10 行目），大小関係を表示する（11 行目）。それ以外の場合，すなわち，c が 0 以上なら，そのまま表示している（13 行目）。

```
1:public class TestIfBlock{
2: public static void main (String argv[]){
3:   int a, b, c;
4:   a = (int)(Math.random() * 10);
5:   b = (int)(Math.random() * 10);
6:
7:   System.out.println("a = " + a + ", b = " + b);
8:   c = a - b;
9:   if(c < 0) {
10       c = - c;
11:      System.out.println("a < b , b - a = " + c);
12:  } else {
13:      System.out.println("a >= b , a - b = " + c);
14:  }
15:}
16:}
```

プログラム５．１　if 文とブロック

if 文のブロックの中で，再び if 文を使用することにより，if 文の入れ子構造を作ることができる。ここでは例題を用いて，if 文の入れ子構造を説明しよう。

ある科目では，テスト，演習，出席点を考慮し，次の方針で成横を決定するとする。

① テストも演習も 80 点以上なら A
② テストが 80 点以上で，演習が 80 点未満なら B
③ テストが 80 点未満で，演習が 80 点以上なら B
④ テストも演習も 80 点未満で，出席が 50 点以上なら C
⑤ テストも演習も 80 点未満で，出席が 50 点未満なら D

この流れを図 5．6 に示す。

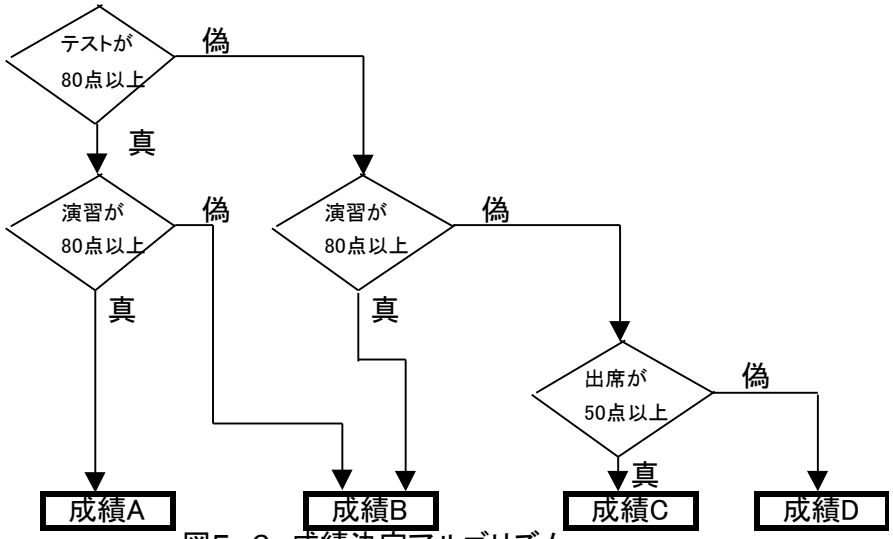

図5．6 成績決定アルゴリズム

このアルゴリズムを実現するプログラムは，次のようになる。

```
//  test：テストの点数，practice：演習の点数，attendance：出席，grade：成績
if（test＞＝80）｛
  if（practice＞＝80）
        grade＝’A‘;
    else
        grade＝’B‘
｝else｛
  if（practice＞＝80）
          grade＝’B‘;
    else｛
      if（attendance＞＝50）
              grade＝’C‘;
          else
              grade＝’D‘;
  ｝
```

}

演習 5．3
　上の成績決定プログラムを完成させなさい。ただし，テスト（test），演習（practice），出席（attendance）の点数は適当に決めてよい。また，点数に対応する成績（grade）が出力されるものとする。

5．6　else－if 文

　前出の例題でも出てきたが，複数の条件判断を必要とする場合が多々ある。このようなときに便利なのが else－if 文であり，次のような形式をとる。

```
if（条件 1）　　文 1；
else　if（条件 2）　文 2；
else　if（条件 3）　文 3；

else　　文 n；
```

　条件 1 が真の場合は文 1 を実行し，偽の場合には条件 2 を調べる。条件 2 が真なら文 2 を実行し，偽ならを条件 3 を調べる。条件が成立するまで次々と調べて行く。プログラムの流れは図 5．7 のようになる。

```
文 0；　　　　　　if 文の前の文
if（条件 1）　　　条件 1 が真ならば
　　文 1；　　　　　　文 1 を実行
else if（条件 2）　条件 1 が偽で条件 2 が真ならば
　　文 2：　　　　　　文 2 を実行
else if（条件 3）　条件 2 が偽で条件 3 が真ならば
　　文 3；　　　　　　文 3 を実行
　...
　else　　　　　　いずれの条件も偽ならば
　　文 n；　　　　　　文 n を実行
　文 n＋1　　　　if 文の後の文
```

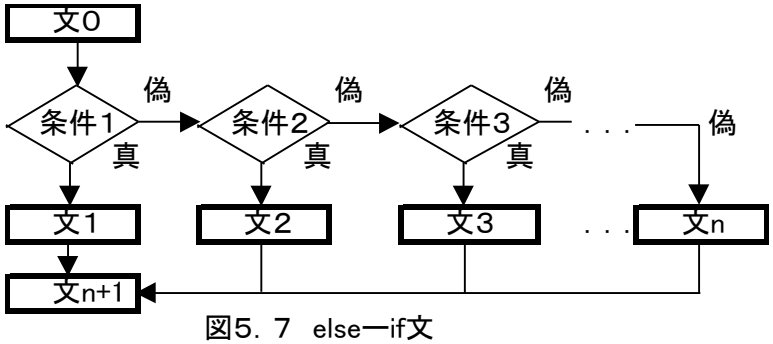

図5．7　else―if文

else－if 文を用いると，演習 5．2 の 2 つの整数の大小判定プログラムを次のように改良することができる。

```
if（a＝＝b）{
        System．out．println（“a＝＝b”）;
} else  if（a＜b）{
        System．out．println（“a＜b”）;
} else {
        System．out．println（” a＞b “）;
}
```

演習 5．4
以下の基準で成績を決定したとする。

A：　80 点以上
B：　60 点以上 80 点未満
C：　40 点以上 60 点未満
D：　40 点未満

int 型で score（点数）を，char 型で grade をそれぞれ宣言し，else－if 文を用いて成績評価プログラムを作成しなさい。score には 0～100 の適当な値を代入しなさい。

5．7　while 文

ある条件を満足している間は，処理を繰り返して実行する，という構造は while 文によって記述できる。while 文は以下のような形式をとる。

```
while（条件）文 ;
```

あるいは，以下に示すように，ブロックを使用して複数の文を繰り返しの対象にすることもできる。

```
while（条件）{文 1 ; 文 2 ; 文 3 ; }
```

プログラムの流れは図 5．8，図 5．9 のようになる。

```
文 0 ;              if 文の前の文
while（条件）         条件が真である限り
      文 1 ;              文 1 を実行
文 2 ;              while 文の後の文
```

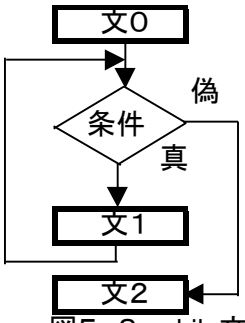

図5．8　while文(a)

```
文0;                    if文の前の文
while(条件)             条件が真である限り
  {文1;文2;～文n;}      文1からnまでを実行
文n+1;                  while文の後の文
```

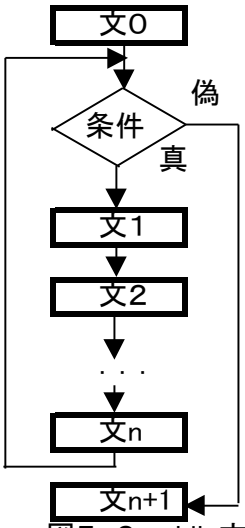

図5．9　while文(b)

通常 while 文の前で条件に関連する変数に値を代入し（初期設定），その変数を使用して条件判定を行う。while 文のブロック内でその変数の値を更新し（再設定），繰り返し実行してよいか条件を検査する。ブロック内で条件に関連する変数の再設定を怠るなど，変数の値を適切に設定しないと，繰り返し実行が終了せず，永遠に while 文から抜けられなくなる。これを無限ループと呼ぶ。
while 文を使用した実例として，1 から 10 までの合計を求めるプログラムを示す。

```
1:public class TestWhile{
2: public static void main (String argv[]){
3:   int i, sum = 0;
4:
5:   i = 1;                  //条件に関連する変数の初期設定
6:   while(i <= 10) {
7:     sum = sum + i;
8:     i ++;                  //条件に関連する変数の値の更新
9:   }
10:
11:   System.out.println("sum = " + sum);
12:}
13:}
```
プログラム５．２　while 文

演習５．５
10 万円を年利 5%の複利で運用したとき，元利合計が 20 万円を超えるのは何年後か。

5．8　for 文

while 文と同様に，繰り返し実行のためにしばしば用いられるのが for 文である。for 文は次のような形式をとる。

for（初期設定；条件；再設定）文（またはブロック）；

for 文のプログラムの流れは以下の通りである。
① 初期設定が行われる（一般には条件に関連する変数の初期設定)。
② 条件の判断が行われる。
③ 条件が満足されたときには続く文やブロックが実行され ④に進む。満足されない場合はループから抜け for 文の次の文に進む。
④ 再設定が行われ ②に戻る。

for 文では，条件の初期設定，判断再設定が 1 つにまとめて記述できるため，while 文と比較してプログラムが読みやすくなる。実際，for 文の形式を while 文で実現すると以下のようになる。

```
初期設定；
while（条件）{
        文；
        再設定；
}
```

プログラム 5．2 を，for 文を用いて書き直してみよう。

```
1:public class TestFor {
2: public static void main (String argv[]){
3:   int i, sum = 0;
4:
5:   for(i = 1; i <= 10; i++)
6:     sum = sum + i;
7:
8:   System.out.println("sum = " + sum);
9: }
10:}
```

プログラム５．３　for 文（TestFor.java）

演習 5．6
10 万円を年利 5%の複利で運用したとき，元利合計が 20 万円を超えるのは何年後か。for 文を用いて求めなさい。

5．9　break 文と continue 文

while 文や for 文での繰り返し実行の途中で，そのループから抜け出すには break 文または continue 文が用いられる。ループのブロックの中で break 文や continue 文が実行されると，ループの実行はそこで中止される。2 つの文の違いは，その後どこから実行を再開するかである。break 文はループそのものを抜けて，ループの次の文から実行する。一方，continue 文はループの先頭に戻る。すなわち，n 回目のループの途中で continue 文が実行されると，n 回目の残りは実行されずに，n＋1 回目の先頭から実行が再開される。

while 文を例に，break 文と continue 文によるプログラムの実行の流れを図 5．10 に示す。

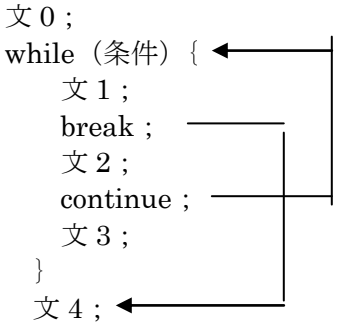

図 5．10　break 文と continue 文のプログラムの流れ

実際のプログラムで break 文と continue 文の働きを比較してみよう。

```
1:public class TestBreak {
2: public static void main (String argv[]){
3:   int i = 0;
4:   while(++i <= 5) {
5:    if(i == 3)   break;
6:    System.out.println(i);
7:   }
8:   System.out.println("The end");
9: }
10:}
```

プログラム５．４　break 文の例（TestBreak.java）

このプログラムを実行すると，i が 3 のときに break 文が実行されて，while 文のブロックから抜け出す。したがって，実行結果は次のようになる。

```
1
2
The end
```

プログラム 5．4 の break 文を，以下のように continue 文に書き換えて実行する。

```
5：　if（i＝＝3）continue；
```

この場合，i が 3 のときに continue 文が実行されて，while 文の条件判断のところに戻る。したがって，実行結果は次のようになる。

```
1
2
4
5
The end
```

これまで，ループは単純なループしか扱ってこなかったが，ループの中にループを入れ子にすることができる。一番内側のループで break 文を実行すると，そのループだけを抜ける。同様に, continue 文の場合も, その文を含んでいる一番内側のブロックの条件部に戻る。しかし，ラベルを使用して，break 文や continue 文でそのラベルを指定すると，実行を再開する場所を制御することができる。

ラベルの形式を以下に示す。

ラベル名：文（またはブロック）；

ラベルは，変数名のように適当な名前にコロン“：”を付けたものである。プログラムの実行上影響はないが，ラベル：に続く文やブロックに名前を付けたことになる。

また，break 文や continue 文で，以下のようにラベルを指定することで，対象となるブロックを指定できる。

break　ラベル名；

continue　ラベル名；

以下の擬似コードでは，一番外側の while 文にラベル“processData”を付けている。while 文に for 文が入れ子になっている。for 文中でエラーが発生すると（変数 error が true の場合），break　processData；が実行される。この break 文は内側の for 文ではなく，processData とラベルが付けられたブロック，すなわち，外側の while 文から抜けることになる。したがって，break 後に実行されるのは文 2 である。

```
processData：while（...）{
    for（...）{
        ...
        if（error）
            break processData；
        ...
    }
    文 1；
}
文 2；
```

5．10　switch 文

switch 文は，n 分岐を実現する構文で次の形式をとる。

```
    switch（式）
   {
        case 定数式 1：　文 1；
        case 定数式 2：　文 2；
```

```
    case 定数式 3 :   文 3 ;
    . . .
    case 定数式 m :   文 m ;
    defau1t :         文 n ;
}
```

ここで式とは，結果が整数になる式[1]である。定数式とは，定数のみの式（正確にはコンパイル時に評価できるもの）で，通常は整数定数や文字定数が使われる。プログラムの流れは図5．11のようになる。

```
文 0 ;                      switch 文の前の文
    switch（式）{
    case  定数式 1 :        式が定数 1 と等しいなら
          文 1 ;                文 1 から実行
    case  定数式 2 :        式が定数 2 と等しいなら
          文 2 ;                文 2 から実行
    . . .
    case  定数式 m :        式が定数 m と等しいなら
          文 m ;                文 m から実行
    default ;               どれとも等しくないなら
          文 n ;                文 n から実行
  }
  文 n＋1 :                 switch 文の後の式
```

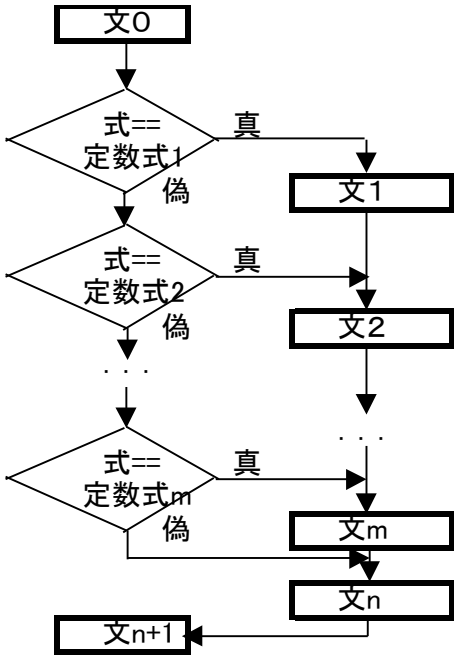

図5．11　switch文

[1] switch 文では char 型，byte 型，short 型，int 型のいずれかが許される。

switch 文は，条件に適合した選択肢の文から始めてそれ以下の文をすべて実行する。実例で確かめてみよう。

```
1:public class TestSwitch {
2: public static void main (String argv[]){
3:   int i = 0;
4:   switch(i + 2) {
5:     case 1:       System.out.println("case 1");
6:     case 2:       System.out.println("case 2");
7:     case 3:       System.out.println("case 3");
8:     default:      System.out.println("default");
9:   }
10:   System.out.println("The End");
11: }
12:}
```

プログラム５．５　switch 文（TestSwitch.java）

このプログラムは，switch 文の式の値は 2 になるので case2：以降が実行され実行結果は次のようになる。

```
case   2
case   3
default
The End
```

n 分岐では，一致した case 文に記述した文だけを実行したいのが普通である。このために，プログラム 5．6 のように各 case 文の処理が終わる場所で break 文を入れる。

```
1:public class TestSwitchBreak {
2: public static void main (String argv[]){
3:   char c = 'a';
4:   switch(c + 1) {
5:     case 'a':       System.out.println("case a");
6:                        break;
7:     case 'b':       System.out.println("case b");
8:                        break;
9:     case 'c':       System.out.println("case c");
10:                       break;
11:    default:        System.out.println("default");
12:                       break;
13:  }
14:  System.out.println("The End");
```

```
15:}
16:}
```
プログラム５．６　switch 文による n 分岐（TestSwitchBreak.java）

プログラム 5．6 では，変数 c の値は，はじめは‘a’だが（3 行目），swicth 文の式は c＋1 なので（4 行目），7 行目の“case　‘b’:...”以降が実行される。8 行目に break 文があるので，そこで switch 文のブロックから抜ける。したがって，実行結果は以下の通りである。

```
case  b
The End
```

switch 文は若干複雑なので，いくつか補足と注意点を挙げておこう。

① default は省略可能
ただし，必要ないときも付けておいた方が分かりやすい。

② 順序は自由
ここでは定数式を昇順に並べたが，順序は自由であり，後で見やすい順序にするべきである。

③ 異なる条件
定数式は，すべて異なる値を持つものでなければならない。

④ break 文
switch 文は，break を付けて使用することが圧倒的に多い。常に break 文を付けることを習慣付けた方がよい。

演習 5．7
演習 5．4 で作成したプログラムに switch 文を加え，点数に相当する成績を表示した後，A は good，B と C は OK，D は bad というメッセージを表示しなさい。