

switch 文...条件分岐処理

三池 克明

switch 文はより複雑な条件分岐ができます。
if 文との違いを理解し、スマートに使い分けられるようにしましょう。

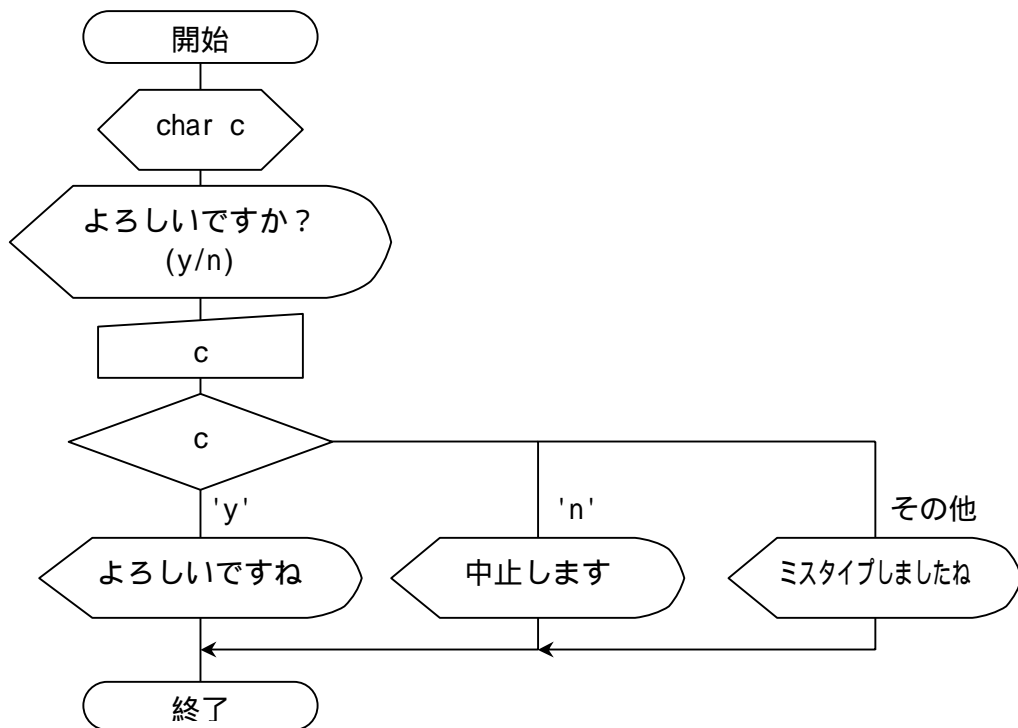
目 次

1.	switch ~ case ~ break.....	1
1.1.	確認を促す...switch 文で記述	1
1.2.	確認を促す...大文字・小文字に対応.....	5
1.3.	確認を促す...よりスマートに.....	6
2.	活用例.....	8
2.1.	メッセージの多分岐.....	8
2.2.	ゲーム風プログラム.....	12
2.3.	式の計算	14
3.	演習問題.....	16

1. switch ~ case ~ break

1.1. 確認を促す...switch 文で記述

以下は if5.cpp と同等の処理を行うプログラムのフローチャートとソースプログラムです。



switch1.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     char    c;
6:
7:     printf("よろしいですか? (y/n) : ");
8:     scanf("%c", &c);
9:
```

```

10:         switch(c){
11:             case 'y':
12:                 printf("よろしいですね。¥n");
13:                 break;
14:             case 'n':
15:                 printf("中止します。¥n");
16:                 break;
17:             default:
18:                 printf("ミスタイプしましたね。¥n");
19:         }
20:     }

```

実行例 1

よろしいですか？(y/n) : y
 よろしいですね。

実行例 2

よろしいですか？(y/n) : n
 中止します。

実行例 3

よろしいですか？(y/n) : a
 ミスタイプしましたね。

それではコードをたどってみましょう。

```

9:
10:         switch(c){
11:             case 'y':

```

10 行目の「switch(c)」は変数 c の値に応じて処理を分岐させることを指示しています。なお stitch() のカッコ内の値は double 型などの実数は使えません。

```
10:         switch(c){
11:             case 'y':
12:                 printf("よろしいですね。¥n");
```

11 行目の「case 'y':」は switch 文で参照した値（この場合は変数 c の値）が 'y'であった場合、次の行から処理をするように指示しています。

よってこのプログラムの場合は 12 行目から処理が続くことになります。

なお、行末の記号はセミコロンではなくコロン「:」であることに注意してください。

```
12:                 printf("よろしいですね。¥n");
13:                 break;
14:             case 'n':
```

13 行目の「break;」は switch 文で囲んだ中カッコ{}から抜けて実行を続けるように指示しています。

よってこのプログラムの場合は 14～19 行目の処理を飛び越えます。

```
14:             case 'n':
15:                 printf("中止します。¥n");
16:                 break;
```

14～16 行目も同じですね。

変数 c の値が 'n'であった場合、「中止します。」のメッセージを表示して、次の break 文で switch 文を抜けます。

```
17:             default:
18:                 printf("ミスタイプしましたね。¥n");
```

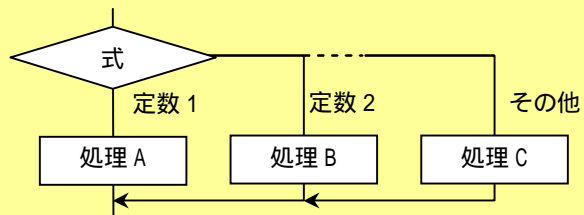
17 行目は case ではなく「default:」となっています。

これは if 文でいう else にあたるもので、「この行以前の case に当てはまらなかった場合」を表します。

また、今回はじめて使用した switch 文の構文は以下のとおりです。

構文：

```
switch(式){  
  case 定数 1:  
    処理 A  
    break;  
  case 定数 2:  
    処理 B  
    break;  
  ...  
  default:  
    処理 C  
}
```



ただし、式や定数は、
int 型などの整数
char 型などの文字
でなければならない

1.2. 確認を促す...大文字・小文字に対応

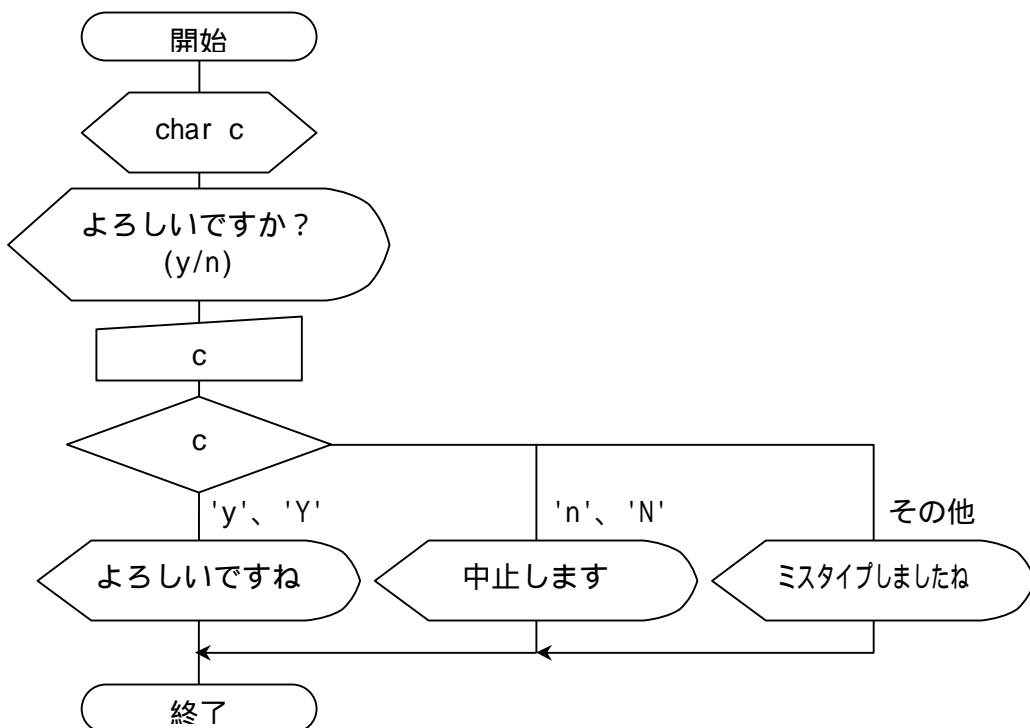
switch1.cpp を改良し、大文字・小文字に対応できるようにしました。
以下のソースプログラムを入力し、コンパイル、実行してみましょう。
(実行例は省略します)

switch2.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     char    c;
6:
7:     printf("よろしいですか?(y/n) : ");
8:     scanf("%c", &c);
9:
10:    switch(c){
11:        case 'y':
12:            printf("よろしいですね。¥n");
13:            break;
14:        case 'Y':
15:            printf("よろしいですね。¥n");
16:            break;
17:        case 'n':
18:            printf("中止します。¥n");
19:            break;
20:        case 'N':
21:            printf("中止します。¥n");
22:            break;
23:        default:
24:            printf("ミスタイプしましたね。¥n");
25:    }
26: }
```

1.3. 確認を促す...よりスマートに

switch2.cpp をさらに改良してみました。
以下がそのフローチャートとソースプログラムです。



swit ch3.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     char    c;
6:
7:     printf("よろしいですか? (y/n) : ");
8:     scanf("%c", &c);
9:
10:    switch(c){
11:        case 'y':
12:        case 'Y':
```

```
13:             printf("よろしいですね。¥n");
14:             break;
15:         case 'n':
16:         case 'N':
17:             printf("中止します。¥n");
18:             break;
19:         default:
20:             printf("ミスタイプしましたね。¥n");
21:     }
22: }
```

実行例 1

```
よろしいですか？(y/n) : y
よろしいですね。
```

実行例 2

```
よろしいですか？(y/n) : n
中止します。
```

実行例 3

```
よろしいですか？(y/n) : a
ミスタイプしましたね。
```

実行例 4

```
よろしいですか？(y/n) : Y
よろしいですね。
```

このように case 文と break 文の性質を利用してコードの行数を減らすこともできます。

複数の case で同じ処理をさせる場合はこのように記述するとよいでしょう。

2. 活用例

多数の分岐処理には if 文よりも switch 文の方が有利です。
本節ではその例を挙げ、解説します。

2.1. メッセージの多分岐

以下は 2004 年 3 月の日を入力し、その日が何曜日なのかを表示するプログラムです。

switch4.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    day;
6:
7:     printf("曜日を知りたい日を入力して下さい\n");
8:     printf("2004/03/");
9:     scanf("%d", &day);
10:
11:    printf("2004/03/%02d は ", day);
12:    switch(day % 7){
13:        case 0:
14:            printf("日曜日です\n");
15:            break;
16:        case 1:
17:            printf("月曜日です\n");
18:            break;
19:        case 2:
20:            printf("火曜日です\n");
21:            break;
22:        case 3:
23:            printf("水曜日です\n");
24:            break;
25:        case 4:
26:            printf("木曜日です\n");
```

```

27:             break;
28:         case 5:
29:             printf("金曜日です\n");
30:             break;
31:         case 6:
32:             printf("土曜日です\n");
33:             break;
34:     }
35: }

```

実行例 1

曜日を知りたい日を入力して下さい

2004/03/5

2004/03/05 は 金曜日です

実行例 2

曜日を知りたい日を入力して下さい

2004/03/10

2004/03/10 は 水曜日です

このプログラムでは日を7で割ったときの余りから処理を分岐させています。では、どうしてこのような方法で曜日を求めることができるのでしょうか。

ここで2004年3月のカレンダーを見てみましょう。

2004年3月						
日	月	火	水	木	金	土
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

続いて日の数字を7で割ったときの余りを求めてみます。

2004年3月の日を7で割ったときの余り

日	月	火	水	木	金	土
	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3	4	5	6
0	1	2	3			

このように余りの数値が曜日ごとに並んでいるのがわかります。

よって7で割った余りが同じ数値なら、それは同じ曜日であることがわかります。

続いて switch4.cpp に手を加えます。

それが以下の switch5.cpp です。

switch5.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    day;
6:
7:     printf("曜日を知りたい日を入力して下さい\n");
8:     printf("2004/03/");
9:     scanf("%d", &day);
10:
11:     printf("2004/03/%02d は ", day);
12:     switch(day % 7){
13:         case 0:
14:             printf("日");
15:             break;
16:         case 1:
17:             printf("月");
18:             break;
19:         case 2:
```

```
20:             printf("火");
21:             break;
22:         case 3:
23:             printf("水");
24:             break;
25:         case 4:
26:             printf("木");
27:             break;
28:         case 5:
29:             printf("金");
30:             break;
31:         case 6:
32:             printf("土");
33:             break;
34:     }
35:     printf("曜日です\n");
36: }
```

このプログラムでは共通部分の処理（「曜日です」の表示）を switch 文の外に記述しました。

今回の場合は若干見づらくなりましたが、この手法は大きなプログラムの開発に有用です。

2.2. ゲーム風プログラム

次のプログラムは昔のコンピュータゲームでよく使われていた処理です。

switch6.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     int    cmd;
6:
7:     printf("どうしますか？\n");
8:     printf(" 1.調べる\n");
9:     printf(" 2.寝る\n");
10:    printf(" 3.移動\n");
11:    printf(" 4.終了\n");
12:    printf("\n 番号を選択してください(1 4):");
13:    scanf("%d", &cmd);
14:
15:    switch(cmd){
16:        case 1:
17:            printf("何も見つからなかった。 \n");
18:            break;
19:        case 2:
20:            printf("あなたはぐっすり寝た\n");
21:            break;
22:        case 3:
23:            printf("あなたはこの場を立ち去った。 \n");
24:            break;
25:        case 4:
26:            printf("終了します。お疲れ様でした。 \n");
27:            break;
28:        default:
29:            printf("その番号は選択できません\n");
30:    }
31: }
```

実行例 1

どうしますか？

1. 調べる
2. 寝る
3. 移動
4. 終了

番号を選択してください(1-4):1

何も見つからなかった。

実行例 2

どうしますか？

1. 調べる
2. 寝る
3. 移動
4. 終了

番号を選択してください(1-4):4

終了します。お疲れ様でした。

メッセージの内容を書き換えるだけでこのようにゲーム的なプログラムにもなります。

2.3. 式の計算

以下のプログラムは2項の式を入力させ計算するプログラムです。

switch7.cpp

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     double a, b, ans;
6:     char op;
7:
8:     printf("<二項演算プログラム> \n");
9:     printf("二項の計算式を入力してください。 \n");
10:    scanf("%lf %c %lf", &a, &op, &b);
11:    switch(op){
12:        case '+':
13:            ans = a + b;
14:            break;
15:        case '-':
16:            ans = a - b;
17:            break;
18:        default:
19:            printf("演算子[%c]が不正です。", op);
20:            return 1;
21:    }
22:
23:    printf("%f %c %f = %f\n", a, op, b, ans);
24: }
```

"%lf %c %lf"
と間に半角スペースを
挿入する

プログラムを
ここで強制終了させる。

実行例 1

<二項演算プログラム>

二項の計算式を入力してください。

10+5.3

10.000000 + 5.300000 = 15.300000

実行例 2

<二項演算プログラム>

二項の計算式を入力してください。

```
10 -      31.4
10.000000 - 31.400000 = -21.400000
```

実行例 3

<二項演算プログラム>

二項の計算式を入力してください。

```
5 -( -3)
5.000000 - 0.000000 = -5.000000
```

このように簡単な式であれば数値と演算子の間に空白が在っても無くても計算してくれます。(さすがにカッコつきの数値はダメでしたが)

このような処理は 10 行目の `scanf()` のように % ~ と % ~ の間に半角スペースを一文字だけ挿入すると可能になります。

```
10: scanf("%lf %c %lf", &a ,&op, &b);
```

半角スペース

半角スペース

これは関数 `scanf()` の仕様です。(その詳細は割愛します)

3. 演習問題

5-1. 「switch4.cpp」は2004年3月の暦に対応しているが、これを改造して今月の暦に対応したプログラム「ensyu5-1.cpp」を作りなさい。

ヒント：

今月の1日は何曜日ですか？

5-2. 「switch7.cpp」は2項の加減算に対応しているが、さらに乗除算にも対応させたプログラム「ensyu5-2.cpp」を作りなさい。なお、演算子はC言語の*や/を使うこと。

ヒント：

演算子が '*' の場合と '/' の場合の処理を追加するだけです。