プログラミング演習 第四回 令和4年10月26日

条件判断 switch 文

if 文は 2 分岐の条件判断。多分岐には switch 文を使うことができる

```
switch(式){
  case 定数1 : 文1;.. ; break;
  case 定数2 : 文2;.. ; break;
  ...
  case 定数n : 文n;.. ; break;
  default : 文n+1;.. ; break;
}
```

式は整数型もしくは文字型

case 定数 : をラベル(名札)という

式を評価し、その値が定数であるラベルの文へ処理が移る。break 文に出会うと switch 文を終了

式の値に合致する定数が無ければ defaultラベルの文に処理が移る（例外処理）

switch 文は、式の値によって処理を複数に分岐

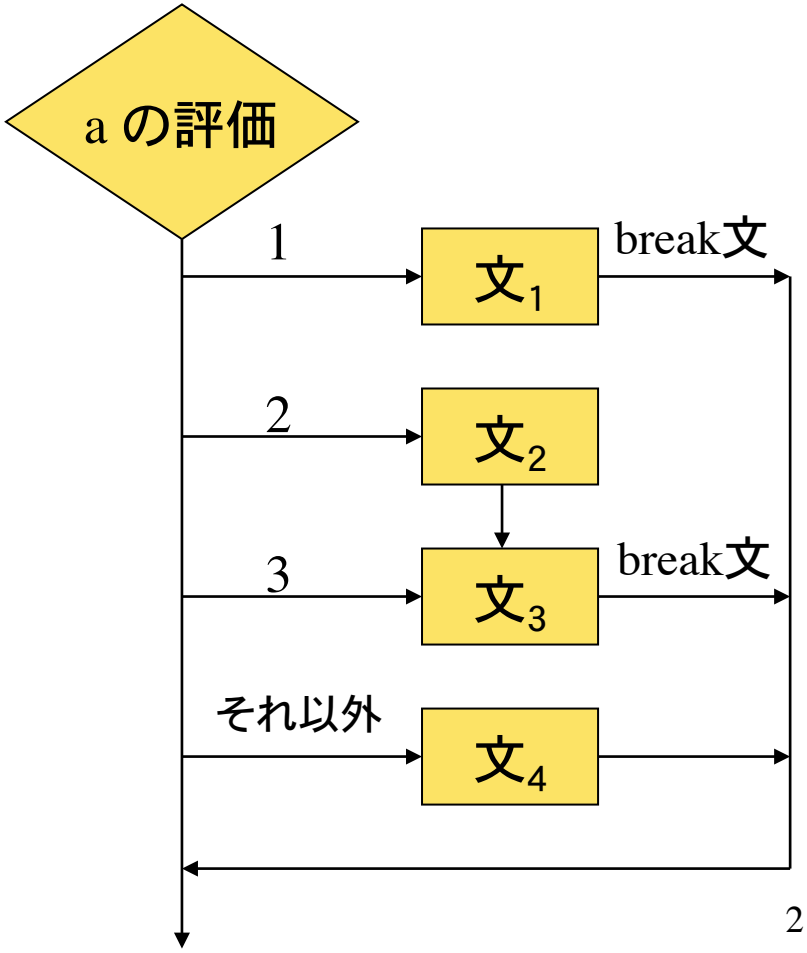
文の後の break 文が無いと分岐後の処理手順が異なる

switch 文と break 文

switch 文は、式の値に対応するラベルへ処理を分岐させる。
break 文は、その文が書かれた処理を終了させる(この場合 switch 文)

```
int a;
scanf_s("%d", &a);

switch(a){
  case 1 : 文1; break;
  case 2 : 文2;
  case 3 : 文3; break;
  default : 文4;
}
```



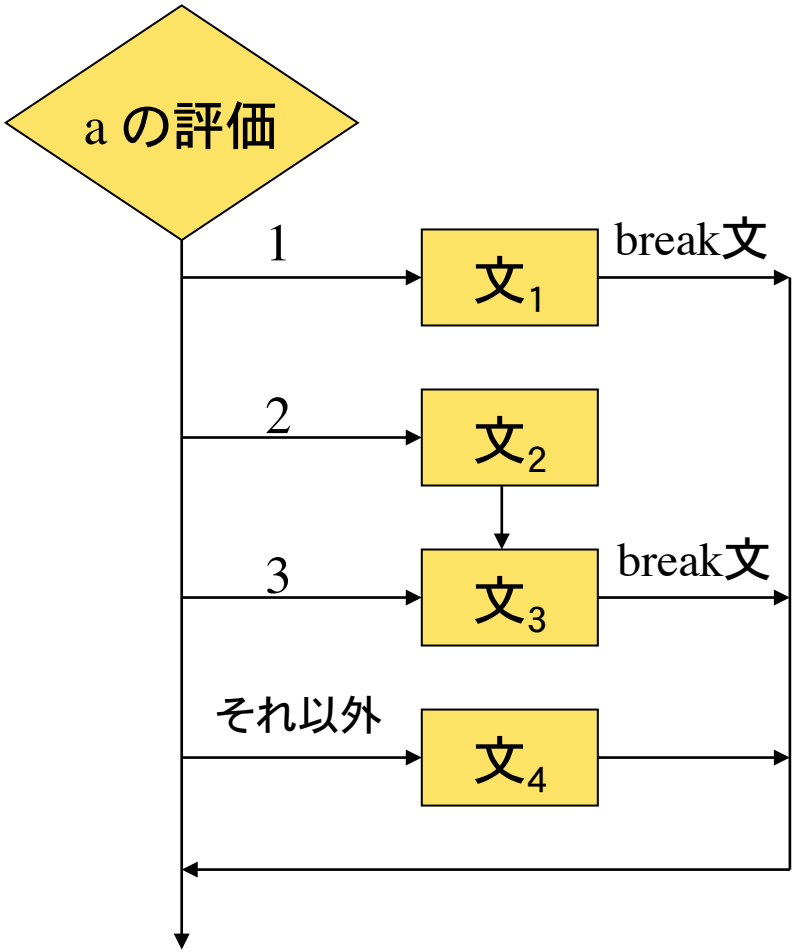
break 文の位置に注意!

break 文を書かないと、次のラベルの行へ処理が移る

一番最後にはbreak文をつけなくともよい。

switch 文と if 文

前項の switch 文



同じことを if 文で書くと、、、

```
int a;
scanf_s("%d", &a);

if( a==1 )
  文1
else
  if( a==2 ){
    文2
    文3
  } else
      if( a==3 )
        文3
      else
        文4
```

switch 文の応用

break 文を意図的に省くことで、多様な分岐処理が可能になる

月（int 1 ~ 12 ）を入力して季節を判断する

```
switch(month){
  case 3:
  case 4:
  case 5: printf("春です\n"); break;
  case 6:
  case 7:
  case 8: printf("夏です\n"); break;
  case 9:
  case 10:
  case 11: printf("秋です\n"); break;
  case 12:
  case 1:
  case 2: printf("冬です\n"); break;
  default: printf("エラーです\n");
}
```

ラベルの後に空文（何もしない文）有り

break 文の位置に注意

defaultラベルで、
例外処理も完璧！

条件判断 if 文と switch 文

2 分岐の条件判断: if 文

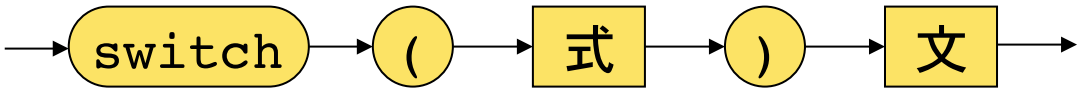
式の真偽で分岐。if 文の入れ子でどのような分岐も可能。

多分岐の条件判断: switch 文

switch 文は、式の値によって分岐先が決まる。
式の値は整数型もしくは文字型に限る。

if 文を switch 文に、switch 文を if 文に書き直すことは可能。しかし、プログラムが読みやすくなるかどうかは別問題。分岐内容によってどちらかを用いる。

switch 文の構文図



文字型

シングルクォーテーション ' で囲った 1 文字を文字定数という

英数文字 A, ?, 1 はプログラム中では 'A' '?' '1' と表記

1 文字を格納する変数の型を文字型といい char で表す

英語で文字を character という

文字型変数の宣言と変数への代入

```
char c;

c = 'A';
```

←―― 変数 c に文字 A を代入

文字定数は文字列リテラルとは異なることに注意!

'A' と "A" は違う

ここでの文字とは英数字および記号を指す。
カナや漢字は除く

文字型変数の入出力

文字型変数の入出力(scanf, printf)の変換指定には `%c` を用いる

```
char a, b;

a = 'A';
scanf_s("%c", &b, 1);

printf("The 1st character is %c\n", a);
printf("The 2nd character is %c\n", b);
```

文字型変数は英数文字 1 文字を格納する。
上のプログラムの scanf _sの入力で、1 文字以上の文字を入力しても、最初の 1 文字だけが変数 c に格納される。

scanf_sで文字の入力を行う時は、文字型の変数名と入力する文字数を指定する。この時の文字数は1である。文字数を指定しない時は1として処理される。

1 文字の入力（getchar, putchar）

1 文字の入力と出力を行う標準ライブラリ関数として
getchar と putchar がある

```
int c;

c = getchar();
putchar(c);
```

← キーボードから 1 文字を読み取って整数型変数 c に代入

← 変数 c を文字として出力

1 文字をなぜ整数型の変数に代入する ???

文字と文字コード

文字(char)は、整数値(int)の文字コードで表される

'0' ゼロ、という文字は、30(16 進数) = 48(10 進数)
'1' 1、という文字は、31(16 進数) = 49(10 進数)

文字と文字コードを対応させる決まりとして、ASCIIコード(JISコード)がある
(ASCII:American Standard Code for Information Interchange)

JIS コード表(16 進数表記)

JIS	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

10 進数と 16 進数

10 進数：'0' ~ '9' の 10個の記号を用いて表記
16 進数：'0' ~ '9' + 'A', 'B', 'C', 'D', 'E', 'F' の 16個の記号を用いて表記

10進数の 100 を16進数で表記すると、

110 = 6 * 16 + 14 であるので、6E

10進数から16進数への変換方法＝16で割り算して商と余り

16進数の 35 を10進数で表記すると、

3 * 16 + 5 = 53 であるので、53

16進数から10進数への変換方法＝16でかけ算した合計

8 進定数：0ではじまる数字の列　　0123
16 進定数：0x または 0Xで始まる数字の列　0x3a

文字型と整数型の関係

文字型は、整数型の部分集合である。

char　　: 0 ~ 255（もしくは -128~127)
int　　: -2147483648 ~ 2147483647

```
char c;
int i;
```

変数 c は文字型として宣言。格納可能な値の範囲は 0 ~ 255。

1 文字を入力する関数 getchar() は、入力時にエラーが発生したり、入力の終わりに達して読むべき文字がない場合には EOF という値を返す（EOF は –1 という整数値）

そのため、getchar() の返却値を格納する変数は整数型でなくてはならない。

```
c = getchar();
```

← EOF を正しく取り扱えないので不可

```
i = getchar();
```

文字＝文字コード（整数値）

```
int c;

c = 100;
printf("%c %d\n", c, c);

c = 101;
putchar(c);

c = 'A';
putchar(c);
printf("%c %d\n", c, c);

c = c + 32;
printf("%c\n", c);

c = ‘a’ - 32;
printf("%c\n", c);
```

100 (10進数) = 64 (16進数) = 'd'

101 (10進数) = 65 (16進数) = 'e'

'A' = 41 (16進数) = 65 (10進数)

'A' +32 = 97 = 61 (16進数) = ‘a’

‘a’ -32 = 65 = 41 (16進数) = ‘A’

大文字と小文字

アルファベット大文字の文字コードの範囲は、41 ~ 5A (16 進数)
A: 16 進数の 41 は、4 * 16 + 1 = 65 (10 進数)
Z: 16 進数の 5A は、5 * 16 + 10 = 90 (10 進数)

アルファベット小文字の文字コードの範囲は、61 ~ 7A (16 進数)
a: 16 進数の 61 は、6 * 16 + 1 = 97 (10 進数)
z: 16 進数の 7A は、7 * 16 + 10 = 122 (10 進数)

```
int c;
c = getchar();
if( c >= 65 && c <= 90 )
  printf("%c は大文字\n", c);
```

```
int c;
c = getchar();
if( c >= 'A' && c <= 'Z' )
  printf("%c は大文字\n", c);
```

```
int c;
c = getchar();
if( c >= ‘a’ && c <= ‘z’ )
  printf(“%c の大文字は %c\n", c, c - 32);
```

問題 1

月（1 から 12）を入力して、四季を判定するプログラムを作成せよ。
条件分岐には、switch文を使うこと。
不適当な入力はその旨表示して処理すること。

```
月を入力せよ: 5
 5 月は春です
```

```
月を入力せよ: 77
 そんな月はありません
```

問題 2

コンピュータとジャイケンをするプログラムを作れ。グーチョキパーをそれぞれ0、1、2の数値で表す。コンピュータの手は乱数を使って生成する。これには、関数rand()を使う。

```
#include “pch.h”
#include <iostream>
#include <time.h>

int main(int ac, char *av[])
{
   int myhand, yourhand;
   srand(time(NULL));
   myhand = rand() % 3;

         ....
}
```

time()関数を使うため必要。

srand()はrand()で生成する乱数の初期値を設定する。ここではtime関数で現在時刻を得る。

rand()で生成した乱数を3で割った余りから、0,1,2のいずれかの数を作りそれをコンピュータの手とする。

```
グーチョキパーのいずれかを入力。
0：グー、1：チョキ、2：パー　：1
あなたはチョキで私はグーです。
私の勝ちです。
```

この色はプログラムによる出力

問題 3

大文字のアルファベット 1 文字を入力して小文字に変換するプログラム

文字コード表を参考にせよ

大文字のアルファベット1文字を入力? V
Vの小文字は vです

大文字のアルファベット1文字を入力? a
aは大文字のアルファベットではありません

ヒント

アルファベットの大文字と小文字は
文字コードで 20(16 進数)= 32
(10 進数)の違いがある

大文字のアルファベットの文字コードの
範囲は、41 ~ 5A(16 進数)、10進数では
65から90。それ以外のコードの文字は
大文字のアルファベットではない