C言語講座第弐回

演算子　分岐　配列　ループ

目次

- 演算子
- 分岐
- ループ
- 配列
- 練習問題

演算子

・前回の四則演算に加えて論理演算子、比較演算子がある。

・論理演算子は集合と同じ。

真(1)と偽(0)の二種類を(!, &&, ||)の組み合わせで表す。

x	y	！x （否定）	x&&y （論理積、かつ）	x||y （論理和、もしくは）
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

・比較演算子は（＜、＞、＝、！）の組み合わせで表す。

・注意点は、“=”だけだと代入になるので、“等しい”という意味で使う場合は“==”を使う。

a < b	aはbより小さい
a > b	aはbより大きい
a == b	aとbは等しい
a <= b	aはb以下
a >= b	aはb以上
a != b	aとbは不等

分岐処理①（if文）

・ifは“もし～ならば”、elseは“それ以外”を意味する。

構文

```
if (式1) {
処理1;
 } // もし、式1が真ならば処理1を行う
else if ( 式2 ) {
処理2;
 } // 式1が偽で、式2が真ならば処理2を行う
else {
処理3;
} // どの処理も行われなければ処理3を行う
```

例文

```
#include <stdio.h>

void main(void)
{

        int num; printf(“整数を入力してください：”);
        scanf("%d", &num);

        if (num % 5 == 0) {

                printf("5で割りきれます¥n", num);
        } else {
                printf("5で割りきれません¥n", num);
        }

}
```

分岐処理②（switch文）

・式の値に応じて処理を切り替える。

If文との違いは条件判断を繰り替えさないので動作が速い。

構文

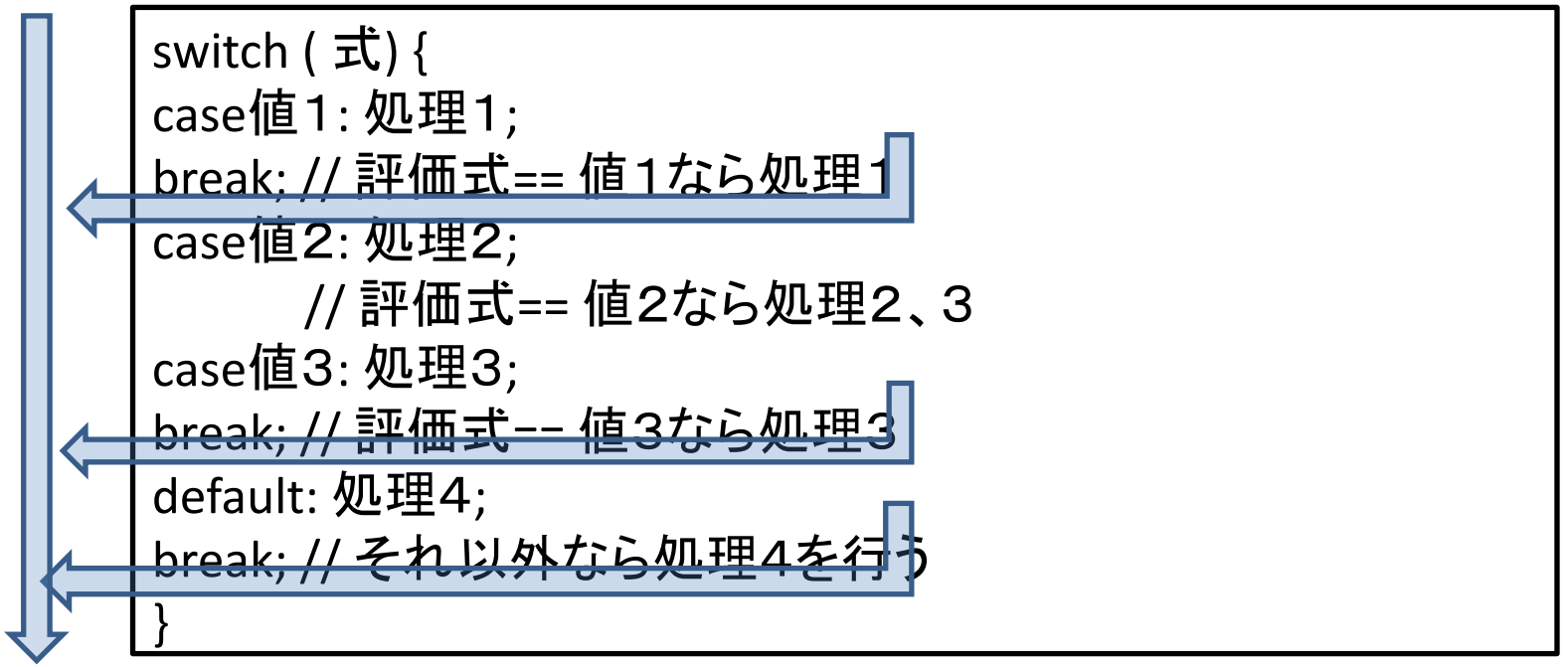
```
switch ( 式) {
case値１: 処理１;
break; // 評価式== 値１なら処理１
case値２: 処理２;
break; // 評価式== 値２なら処理２
case値３: 処理３;
break; // 評価式== 値３なら処理３
default: 処理４;
break; // それ以外なら処理4を行う
}
```

例文

```
#include <stdio.h>
int main(void){
        int input;
        printf( “数字を入力して下さい¥n" );
        scanf( "%d", &input );
        switch( input ){  /* 入力された文字に応じた分岐 */
        case 1:   /* 1のとき */
                  printf( “1が入力されました¥n" );
                  break;
        case 2:   /* 2のとき */
                  printf( “2が入力されました¥n" );
                  break;
        default:    /* 1でも2でもないとき */
                  printf( “1,2以外の数字が入力されました¥n" );
                  break;
        }
        return 0;
}
```

・注意点は処理の後に“break”をつける。

処理の後にこれが無いと意図しない処理まで行ってしまう。



繰り返し文

- 似た処理を何度でも呼び出すことが出来る。
- 繰り返し文にはdo-while文,while文,for文がある。

構文（do−while文）

```
do {
処理; //条件を満たす間、繰り返す
} while (条件);
```

構文（while文）

```
while (条件) {
処理; //条件を満たす間、繰り返す
}
```

do-while文とwhile文の違い

do-while文

```
#include <stdio.h>
void main(void){
int num= 0;
do {
printf("do-while¥n");
} while (num == 1);
}
```

while文

```
#include <stdio.h>
void main(void){
int num= 0;
while (num == 1) {
printf("while¥n");
}
}
```

do-while文はループ内の処理の後で条件式の判定を行う
while文はループ内の処理の前に条件式の判定を行う

for文

- ループの回数を指定できる

構文

```
for (初期化式;条件式;カウンタ) {
処理 ; //条件を満たす間、繰り返す
}
```

break文

- ループ処理を抜け出せる。
- 抜け出せるのはbreak一つにつきループ一つ

例文

```
#include <stdio.h>
void main(void)
{
int num = 0;
while ( 1 == 1 ) {
if (num >= 10) { break; }                    //10以上だとbreak
printf("%d ", num);
num += 1;
}
}
```

continue文

・breakは完全にループ処理を抜けるのに対し、continueは一時的に処理を抜かすために使う。

例文

```
#include <stdio.h>
void main(void)
{
int num = 0;
while ( num < 9 ) {
num += 1;
if (num % 2 == 1) { continue; }              //余りが1だとprintfが行われない
printf("%d ", num);
}
}
```

配列

・同じデータ型を格納する変数をまとめて管理することができる
要素の番号は必ず0から始まる

構文

データ型配列(変数)名[要素数]

例文

```
int num[3];
num[0] = 1;
num[1] = 2;
num[2] = 3;
```

多次元配列

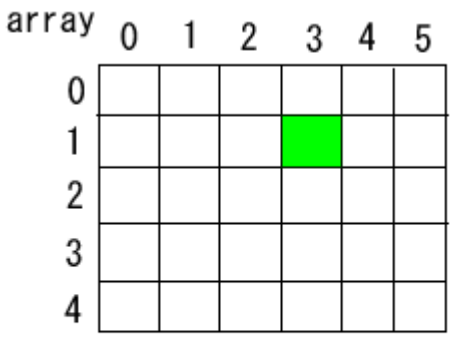
- 二次元配列は、各要素が縦横にならんでいるもの

構文

```
データ型 配列[縦の要素数][横の要素数];
```

例文

```
int array[5][6];
array[0][4] = 1;
array[1][3] = 2;
array[2][2] = 3;
```



緑の部分は、

array[1][3] となる

(*M*)練習問題(*´ω`*)

①5つの要素を持つint型配列を用意し、その全ての要素をその要素番号に初期化して表示するプログラムを作成してください。

a[0] = 0、a[1] = 1、a[2] = 2、a[3] = 3、a[4] = 4

初期化する部分はwhile文で。

表示する部分はfor文で。

②

scanfでテストの点数(0-100)を入力してもらい、その結果によって評価を表示するプログラムを作成してください。

評価は

90-100：A+　　80-89 :A

70-79 :B　　60-69 :C

0-59 :論外

それ以外：・・・ねーよwww

回答

①

```
#include <stdio.h>
void main(void){
int array_Int[5]; int i,j;
i = 0;
while (i < 5) {
array_Int[i] = i;
i++;
}
for (j = 0; j < 5; j++) {
printf("%d¥n", array_Int[j]);
}
}
```

②

```
#include <stdio.h>
void main(void) {
int point;
printf("0-100の点数を入力してください:");
scanf("%d", &point);
        if (point >= 0 && point <= 100) {
                if (point >= 90) {
                        printf("A+¥n");
                } else if (point >= 80) {
                        printf("A¥n");
                } else if (point >= 70) {
//次ページへ
```

```
                printf("B¥n");
            } else if (point >= 60) {
                printf("C¥n");
            } else {
                printf("論外¥n");
            }
        } else {
            printf(“・・・ね―よwww¥n");
        }
}
```