C 言語 012 配列

【配列】

配列はひとつの変数で複数の値が記録できるものです。

ひとつの変数を複数に分け、分けられた物を「要素」と称します。

要素には添字（番号）が割り振られ、添字は 0 から始まります。

＜配列の宣言＞

```
データ型 配列名[ 要素数 ];
```

＜値を代入する＞

```
配列名[ 添字 ] = 値;
```

＜例文＞

```
int array[10];
array[5] = 10;
```

配列 array は 0～9 までの 10 個に分けられ、5 番目の要素に 10 が代入されます。

配列 array の値

添字	0	1	2	3	4	5	6	7	8	9
値						10				

＜配列の初期化＞

```
int array[10] = {9, 8 , 7, 6, 5, 4, 3, 2, 1, 0};
もしくは
int array[] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};    /* 自動的に要素数は 10 個になる */
```

配列 array の値

添字	0	1	2	3	4	5	6	7	8	9
値	9	8	7	6	5	4	3	2	1	0

配列の要素も初期化していない場合は不定値になります。

＜要素数に注意＞

```
int array[10];
array[10] = 10;
```

このコードはコンパイルエラーにはなりませんが、実行しても正常に動作しません。

array の要素数は 10 個、添字は 0～9 になり、array[10]は存在しないからです。要素数には注意。

【配列の要素数を取得する】

sizeof 演算子を利用します。sizeof 演算子は変数のサイズ（バイト数）を取得する演算子です。

<size_t 型>

バイト数を記録するためのデータ型で、コンパイラによって異なりますが、MinGW では

「typedef unsigned long int size_t;」で定義されています。

「stddef.h」に定義されていますが、「stdio.h」「stdlib.h」をインクルードしても使用できます。

＜例文 1＞

```
int var;
size_t len = sizeof(var);    /* int 型は 4 バイト、変数 len には 4 が代入される */
```

＜例文 2＞

```
int array[5];
size_t len = sizeof(array) / sizeof(array[0]);    /* 変数 len には 5 が代入される */
size_t len = sizeof(array) / sizeof(int);         /* この書き方でも可 */
```

「sizeof(array)」は配列 array のサイズを取得しています。int 型（4 バイト）×要素数 5＝20 バイトになります。

「sizeof(array[0])」は要素 1 個分のサイズを取得しています。配列 array は int 型なので 4 バイトになります。

20÷4＝5 で、要素数を取得できます。

【for 文との組み合わせ】

for 文と合わせて使用すると、効率良く配列を処理できます。

```
int array[10];
size_t len = sizeof(array) / sizeof(int);
int i;
for (i = 1; i < len; i++) {    /* i が 10 を超えるとエラーになるので注意 */
    array[i] = 9 * i;
}
```

array に九九の 9 の段を代入しています。

配列 array の値

添字	0	1	2	3	4	5	6	7	8	9
値		9	18	27	36	45	54	63	72	81

【配列を関数の引数にする】

＜配列を引数にする場合の例文＞

```
/* メイン関数 */
int main(void) {

    /* 関数プロトタイプ宣言、データ型に[]を付けて配列を指定 */
    void print_array(int[]);

    /* 配列作成 */
    int array[] = {1, 2, 3, 4, 5};

    /* 配列名を引数に指定 */
    print_array(array);

    return 0;
}

/* 関数の記述 */
void print_array(int array[]) {       /* 引数を配列にする */
    int i;
    for(i = 0; i < 5; i++){
        printf("%d¥n", array[i]);     /* 順番に要素の値を出力 */
    }
}
```

配列を関数の引数にした場合は「参照渡し」になります。

＜配列を戻り値にすることはできない＞

C 言語では関数の戻り値に配列を返すことはできません。「参照渡し」で代用します。

【多次元配列】

配列は多次元化することができます。

```
int array[2][3];    /* 二次元配列 */
array[0][1] = 10;
array[1][1] = 20;
```

配列 array は 2×3＝6 の二次元配列になります。

配列 array の値

	0	1
0		
1	10	
2		20

← 1 個目の要素

↑ 2 個目の要素

＜初期化＞

```
int array[2][3] = { {0, 1, 2}, { 3, 4, 5} };
もしくは
int array[][3] = { {0, 1, 2}, { 3, 4, 5} };    /* 省略できるのは最初の要素数だけ */
```

＜三次元配列＞

```
int array[2][3][4];    /* 三次元配列 */
array[0][1][2] = 10;
```

配列 array は 2×3×4 の三次元配列になります。

配列 array の値

1 個目の要素[0]

	0	1	2
0			
1			
2		10	
3			

1 個目の要素[1]

	0	1	2
0			
1			
2			
3			