

2011 年度 情報科学&情報科学演習 ～C 言語のまとめ～

2011 年 6 月 22 日

■ データ型：数値や文字を扱うために変数や配列 (文字列) を定義する。

データ型	意味	データの値の例
char	1 文字 (character)	'a', 'b', 'c', ... (「'」で囲む) 97, 98, 99, ... (数値で指定)
int	整数 (integer)	..., -2, -1, 0, 1, 2, 3, ... (10 進数 (decimal)) * 8 進数 (octal) は先頭に「0」を付加 * 16 進数 (hexadecimal) は先頭に「0x (or 0X)」を付加
double	倍精度 浮動小数点数 (double float)	0.00000123, 333.3 (小数表現 (float)) 1.23e-6, 3.333E+2 (指数表現 (exponent))

表 1: 主なデータ型の種類

配列：同じデータ型の変数を連続してメモリ上に確保した変数の列 (下図参考)。

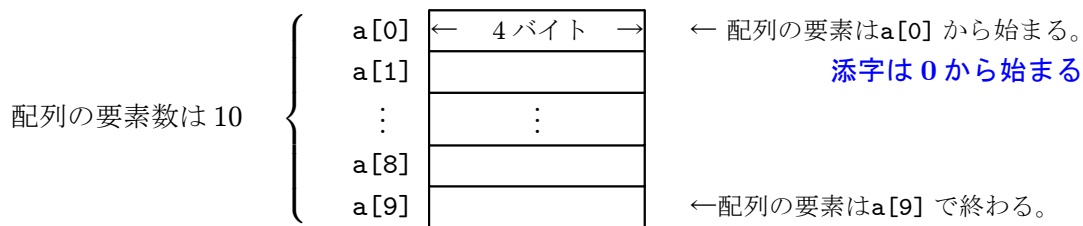


図 1: 配列「int a[10]」の構造

配列 (変数) の宣言では配列長は具体的な数値で指定すること。

* 初期値を設定する場合は最初の添字のみ省略可。

- 1 次元配列の宣言は「データ型 変数名 [配列長]」
「double a[10]», 「int a[3] = {1, 2, 3}», 「int a[] = {1, 2, 3}」
- 2 次元配列の宣言は「データ型 変数名 [配列長] [配列長]」
「double a[10][20]», 「int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}」
- 文字列 (**string**) の宣言は「char 変数名 [配列長]」 (**終端を示す「'\0」を常に考慮**)
「char s[80]», 「char s[4] = "abc"», 「char s[] = {'a', 'b', 'c', '\0'}」
「char s[8] = "abcde"」と宣言した場合：

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]
'a'	'b'	'c'	'd'	'e'	'\0'	(未定)	(未定)

配列の要素を参照または変更するには、配列の位置を示す添字を指定する (**添字は0から始まる**)。

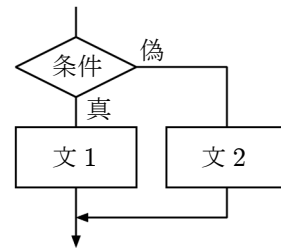
例えば「int a[5] = {1, 2, 3, 4, 5}」と宣言されている場合：

- 「1」を参照するには「a[0]», ..., 「5」を参照するには「a[4]」とそれぞれ記述する。
- 配列中の「3 (=a[2])」を「6」に変更するには「a[2] = 6」と記述する。

■ 制御文：プログラム処理の流れを制御する。

if 文の書式：

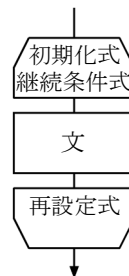
```
if ( 条件 ) {  
    文 1  
}  
else {  
    文 2  
}
```



- if 文は 2 方向分岐を行なう。
- 「条件 (制御式)」が真ならば「文 1」を、偽ならば「文 2」を実行する。
- else 以下は不要なら省略できる。

for 文の書式：

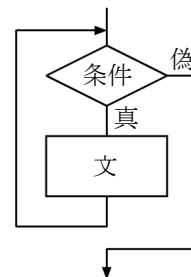
```
for ( 初期化式 ; 継続条件式 ; 再設定式 ) {  
    文  
}
```



- for 文は指定回数反復処理を行なう。
- まずループ処理に入る前には「初期化式」を実行する。
- その後、「継続条件式 (制御式)」が真の間、「文」を実行する。
- 各ループ処理のたびに文の実行後、「再設定式 (変数の増減)」を実行する (インクリメント演算子「i++」($\Leftrightarrow$ 「 $i = i + 1$ 」)やデクリメント演算子「i--」($\Leftrightarrow$ 「 $i = i - 1$ 」)がよく使われる)。

while 文の書式：

```
while ( 条件 ) {  
    文  
}
```



- while 文は不定回数のループ処理を行なう (前判定)。
- 「条件 (制御式)」が真である間、「文」を繰り返し実行する。ただし、はじめから偽であれば「文」は 1 度も実行されない。
- while 文は次のループに入る「条件」を判断する。

■ 入力関数 printf() と出力関数 scanf() : データの入力と出力を行なう。

printf() 関数の書式 :

```
printf(" 書式指定文字列 ", データの並び );
```

- 「書式指定文字列」は、一般文字列・変換仕様 (%ではじまる)・エスケープシーケンス (\ではじまる) を用いて記述する。

「printf("%f+%f=%f\n", a, b, a + b);」, 「printf("This is the %s.\n", s);」

- 変換仕様のオプションの書式 :

% フィールド幅 . 精度 変換指定子

* フィールド幅指定子 : 数値の出力幅を指定する。ただし、必要幅は確保され出力される。

* 先行0 指定 : 桁数が足りない場合に先行する空白部分を0 で埋める。

* 精度指定子 : 浮動小数点数の小数点以下の桁数を指定する。

scanf() 関数の書式 :

```
scanf(" 書式指定文字列 ", 変数アドレスの並び );
```

- 「書式指定文字列」は、一般文字列・変換仕様 (%ではじまる)・エスケープシーケンス (\ではじまる [\nは使用不可]) を用いて記述する。ただし、変換仕様以外の部分は捨てられる。

「scanf("%lf,%lf", &a, &b);」, 「scanf("gcd(%d,%d)", &c, &d);」

- 「変数アドレスの並び」の変数名の前には、アドレス参照を表す「&」を付加する。

ただし、文字列は、最初からアドレスによる参照なので付加しない。

例えば、文字列を読み込む場合は「scanf("%s", s);」のように記述する。

変換仕様とエスケープシーケンス (まとめ) :

変数の種類	printf() 関数	scanf() 関数
char 型	%c	%c
文字列	%s	%s
int 型	%d	%d
double 型	%f, %e	%lf

表 2: 変換仕様

記号	意味
\n	改行 (new line) を出力 (scanf() では使用不可)
\t	タブ (tab) を入力または出力
\\	自身を入力または出力
\"	"を入力または出力
%%	%を入力または出力 ← これは変換仕様

表 3: エスケープシーケンス

■ 関数：C言語によるプログラミングは、関数を記述することである。

関数の基本形：

```
戻り値の型 関数名 ( 引数の宣言, ... )
{
    変数宣言と文を記述

    return 戻り値;
}
```

● ユークリッドの互除法を用いて3変数の最大公約数を求めるプログラム

[illegible]

第 14 行の変数 `x`, `y` は関数 `gcd()` のローカル変数で、関数 `main()` の変数 `x`, `y` とは関係ない。従って、以下のように記述してもよい (独立したローカル変数であることがよくわかる)。

```

14: int gcd(int a, int b)           ← 戻り値の型 int, 引数の宣言 int a, b
15: {
16:     int tmp;                    ← ローカル変数の宣言 int tmp
17:
18:     while (b != 0) {
19:         tmp = a % b;             ← a を b で割った余り a % b
20:         a = b;
21:         b = tmp;
22:     }
23:
24:     return a;                   ← 関数 gcd() の戻り値 a
25: }

```