C 言語 003 変数

【変数】

変数はメモリ内でデータ＝値を一時的に記録する容器のような物です。
ユーザが入力した値を記録したり、演算結果を記録したり、プログラムを実行する上で必要な値を記録するのに使用します。

＜変数の基本的な使い方＞

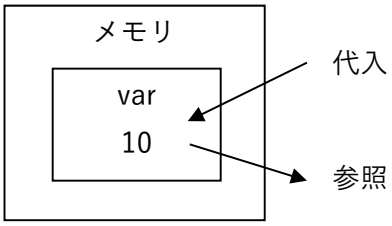
```
#include <stdio.h>
int main(void) {

    /* 変数宣言 */
    int var;

    /* 代入 */
    var = 10;

    /* 参照 */
    printf("%d¥n", var);    /* 10 と出力 */

    return 0;
}
```



変数の基本的な使い方は、変数を使用するための「変数宣言」、変数に値を入れる「代入」、変数に入れた値を使用する「参照」の 3 つです。

＜変数宣言＞

```
データ型 変数名;
```

「データ型」は記録する値が数値なのか文字なのか、何の値を記録するのかを指定するものです。
「変数名」の名付け方には下記のルールがあります。

- 文字数は 31 文字以下。
- 変数名で使用できる文字は「英語」「数字」「_（アンダーバー）」で、頭文字に「数字」は使用できない。
- 大文字と小文字は別文字として扱われる。
- 変数名は重複してはいけない。
- C 言語で定義されている「予約語」は使用できない。
  予約語は特殊な用途で用いられる単語ゆえ、変数名には使用できません。

【C 言語のデータ型】

データ型		サイズ（バイト）	記録できる値
整数型	signed short int（short）	2	-32768～32767 の整数
	unsigned short int	2	0～65535 の整数
	signed long int（long）	4	-2147483648～2147483647 の整数
	unsigned long int	4	0～4294967295 の整数
	signed int（int）	short、long の指定が無い場合はコンパイラによって short 型か long 型になり、MinGW では signed long int 型になる。	
	unsigned int	MinGW では unsigned long int 型になる。	
浮動小数点型	float	4	1.17549E-38～3.40282E+38 の実数
	double	8	2.22507E-308～1.79769E+308 の実数
文字型	signed char（char）	1	-128～127 までの整数か 文字（ASCII コード）1 バイト
	unsigned char	1	0～255 までの整数か 文字（ASCII コード）1 バイト

- 「signed」は符号有り（負数有り）、「unsigned」は符号無し（負数無し）になる。
- 「signed」「int」の記述は省略できる。「signed short int」は「short」、「signed long int」は「long」、「signed int」は「int」と省略できる。
- 実数は整数と小数のこと。
- char 型は整数型であり、文字データ（ASCII コード）を整数データで記録している。
- コンパイラによっては long int よりも多くの整数が記録できる「long long int」や、double よりも多くの実数が記録できる「long double」がある。
- 一般的に整数型は int、浮動小数点型は double、文字型は char が使用される。

【宣言部と処理部】

C 言語のソースコードは「宣言部」と「処理部」に分けられ、変数宣言は宣言部（ブロックの最初）に記述し、処理部よりも先に記述しないといけません。

```
#include <stdio.h>
int main(void) {

    /* 宣言部は処理部よりも先に記述すること */
    int var;

    /* 処理部  */
    var = 10;
    printf("%d¥n", var);
}
```

【変数への値の代入】

変数に「=」（代入演算子）を使用して値を代入します。

整数型	var = 10;	var の中に整数 10 を代入。 8 進数で代入したい場合は先頭に 0 を付ける。例：010 16 進数で代入したい場合は先頭に 0x を付ける。例：0x10
浮動 小数点型	var = 3.14;	var の中に実数 3.14 を代入。
文字型	var = 'a';	var の中に文字「a」を代入。 文字型は値の前後を'（シングルクォーテーション）で括る。

- 整数型の変数に小数を代入すると小数部は切り捨てられ整数になる。
- 記録できる範囲を超える値を代入した場合、例えば short 型の変数に 32767 よりも大きい整数を代入した場合、「オーバーフロー（桁あふれ）」が発生し、値がおかしくなる。MinGW ではコンパイル時に警告が出力される。

【初期化】

変数に最初に代入する値を「初期値」、初期値を代入することを「初期化」と称します。
変数は宣言時に初期化ができます。

＜変数宣言時に初期化＞

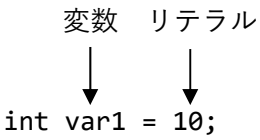
```
データ型 変数名 = 値;
```

＜例文＞

```
int  var1 = 10;      /* 変数 var1 を宣言すると同時に 10 を代入 */
char var2 = 'a';     /* 変数 var2 を宣言すると同時に a を代入 */
```

変数を宣言しただけで初期化していない場合、変数の値は不定値（メモリの状態によって異なる値）になります。

【リテラル】



var1 は「変数」です。var1 に 10 を代入すれば var1 は 10 に、var1 に 100 を代入すれば var1 は 100 に、代入した値によって変わるので「変数」です。
10 は「定数」もしくは「リテラル」と称します。当たり前なことですが、10 と記述すれば 10、100 と記述すれば 100 になり、変動しない値なので「定数／リテラル」です。

C 言語では整数リテラルは「int 型」、実数リテラルは「double 型」として扱われます。
他のデータ型を指定したい場合はリテラルに接尾語（末尾に付ける句）を付けます。

データ型	接尾語	例文
unsigned int	u、U	10U
signed long int	l、L	10L
unsigned long int	ul、UL	10UL
float	f、F	10.5F

【const 変数】

「const」を付けて宣言した変数を「const 変数」と称します。

const 変数は「値を変更できない変数」で、変数宣言時に必ず初期化する必要があり、初期値を変更できません。

```
const int var = 10;    /* const 変数は必ず初期化すること */
var = 20;              /* 値を変更できずエラーになる */
```

const 変数を使用した円の面積を演算する例文

```
#include <stdio.h>
int main(void) {

    /* 変数宣言 */
    const double pi = 3.14;    /* 円周率は 3.14 で固定 */
    double radius;             /* 半径 */
    double area;               /* 円の面積 */

    /* 半径の入力 */
    printf("半径を入力して下さい。¥n");
    scanf("%lf", &radius);

    /* 円の面積の演算 */
    area = radius * radius * pi;

    /* 円の面積を出力する */
    printf("面積は%f¥n", area);

    return 0;
}
```

変数 pi は初期値を変更できません。

【typedef】

typedef を使用すればデータ型の名前を変えることができます。

＜typedef の宣言＞

```
typedef 既存のデータ型名 新しいデータ型名;
```

＜例文＞

```
typedef unsigned long int num;    /* unsigned long int 型を num 型と定義する。 */
num var1 = 10;                    /* num 型（実際は unsigned long int 型）の変数 */
num var2 = 20;
```

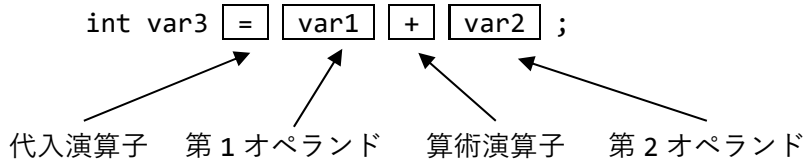
「typedef unsigned long int num;」を「typedef signed short int num;」に変更すれば、変数 var1 も変数 var2 もまとめて「signed short int 型」に変更できます。

【数値の演算】

変数は演算に使用できます。

```
int var1 = 10;
int var2 = 5;
int var3 = var1 + var2;    /* var3 には 15 が代入される */
```

C 言語では「オペランド（演算対象物）」と「演算子」の組み合わせで演算されます。



＜算術演算子＞

演算子	例文	説明
+	var1 + var2	var1 に var2 を足し算する。
-	var1 - var2	var1 に var2 を引き算する。
*	var1 * var2	var1 に var2 を掛け算する。
/	var1 / var2	var1 に var2 を割り算する。
%	var1 % var2	var1 を var2 で割った余りを演算する。 「5 % 3」で 5 を 3 で割った余り 2 が算出される。

＜インクリメント、デクリメント＞

「変数名++」もしくは「++変数名」はインクリメントと称し、変数に+1 します。

「変数名--」もしくは「--変数名」はデクリメントと称し、変数に-1 します。

インクリメント、デクリメントには、変数の前に++や--が前に付く「前置」と、後ろに付く「後置」と 2 通りの記述があります。

前置は処理前に演算され、後置は処理後に演算されます。

```
/* printf される前に+1 される */
int var = 10;
printf("%d¥n", ++var);    /* 11 と出力 */
```

```
/* printf された後に+1 される */
int var = 10;
printf("%d¥n", var++);    /* 10 と出力 */
```

＜演算の優先順位＞

台形の面積の求め方は「(上底＋下底）×高さ÷２」

C 言語も算数と同様、足し算、引き算よりも掛け算、割り算の演算が優先されるので、優先順位を変えたい場合は()を使用します。

```
int above = 10;     /* 上底 */
int under = 20;     /* 下底 */
int height = 30;    /* 高さ */
int area;           /* 面積 */
area = (above + under ) * height / 2;    /* 演算 */
```

【文字と文字列】

プログラムでは「文字データ＝1 文字」「文字列データ＝1 文字以上の文字」としています。

文字データを記録するのには「char 型」、文字列データを記録するには「文字配列」を使用します。

C 言語では文字リテラルは前後を'（シングルクォーテーション）で、文字列リテラルは前後を"（ダブルクォーテーション）で括ってください。

```
printf("%c¥n", 'a');      /* 文字「a」を出力する */
printf("%s¥n", "abc");    /* 文字列「abc」を出力する */
```

なお、リテラルの前後を括る記号のことを「引用符」と称します。

【予約語一覧】

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

【補足　2 進数とデータ型】

コンピュータは情報を 0 と 1 の 2 進数で管理しており、2 進数の 1 桁を「ビット」と称し、データの最小単位となっています。

8 ビット＝1 バイトと定義されており、主にハードウェアではビット、ソフトウェアではバイトが単位として用いられます。

＜整数型＞

例えば、4 ビットでは 0000～1111 まで 16 パターンあり、10 進数の整数だと 0～15 を記録できます。

MinGW の int 型は 4 バイト＝32 ビットなので 4294967296 パターン、10 進数整数 0～4294967295 となります。

ただし、これは「unsigned int 型」の話で「signed int 型」では、最初の 1 ビットが 0＝正数、1＝負数を示す「符号」になるので、31 ビットの 2147483648 パターン、10 進数整数-2147483648～+2147483647 となります。

＜浮動小数点型＞

実はコンピュータでは小数を記録できません。整数を組み合わせて記録します。

浮動小数点は、小数を「仮数」「基数」「指数」の組み合わせで記録する手法で、多くのプログラミング言語やデータベース等で使用されています。

浮動小数点は「仮数 × 基数の指数 乗」で記録します。

10 進数の浮動小数点の場合

1234567890 ＝ 1.23456789 × 10 の 8 乗　（1.23456789E+8 と表現する）

0.123456789 ＝ 1.23456789 × 10 の-1 乗　（1.23456789E-1 と表現する）

10 進数なので基数は 10。仮数は 1.23456789。指数が 8 や-1 になります。

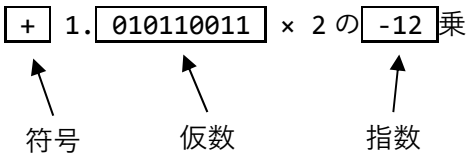
2 進数の浮動小数点の場合

10101100110000000000 ＝ 1.010110011 × 2 の 20 乗

0.10101100110000000000 ＝ 1.010110011 × 2 の-12 乗

2 進数なので基数は 2。仮数は 1.010110011。指数が 20 や-12 になります。

C 言語で浮動小数点を記録する場合、2 進数なので基数は 2 で固定。
「正数か負数を示す符号」「仮数の小数部分」「指数」の 3 つの整数を 2 進数で記録することになります。



各データ型の構成は下記の通り

float（4 バイト＝32 ビット）は符号 1 ビット、仮数＝23 ビット、指数＝8 ビット

double（8 バイト＝64 ビット）は符号 1 ビット、仮数＝52 ビット、指数＝11 ビット

＜文字型＞

C 言語では文字データを「ASCII コード」なる文字コードで管理、記録しています。

ASCII コードは数字、英語、記号等を文字コード 0～127 までの 128 文字で構成されています。

```
char var = 'A';    /* 実際は 65 を代入している */
```

ASCII コードの一部

コード	文字	コード	文字	コード	文字	コード	文字
65	A	72	H	79	O	86	V
66	B	73	I	80	P	87	W
67	C	74	J	81	Q	88	X
68	D	75	K	82	R	89	Y
69	E	76	L	83	S	90	Z
70	F	77	M	84	T		
71	G	78	N	85	U		

日本語を使用する場合は char 型変数を 2 つ使用した 16 ビットで管理しますが、詳しくは「文字配列」で学習します。