第 2 章　変数と演算

本章では、C 言語における変数と、その周辺の知識を説明します。

2.1　変数とは

変数とは、数字や文字などのデータを格納しておく、データの入れ物のようなものです。さまざまな種類の変数を扱うために、さまざまな変数の型（type）が用意されています。代表的な変数の型は、以下のとおりです。

型名	データの種類	扱える範囲
int	整数値	−2147483648 ～ 2147483647
long	倍長整数値	−9223372036854775808 ～ 9223372036854775807
float	実数値	有効数字 7 桁 $\pm 10^{-38}$ ～ 10^{38}
double	倍精度実数値	有効数字 15 桁 $\pm 10^{-308}$ ～ 10^{308}
char	文字	ASCII 文字（−128 ～ 127）

他にもさまざまな型がありますが、とりあえずは

「整数を扱いたいなら int」

「実数を扱いたいなら double」

「文字を扱いたいなら char」

の三つだけを覚えておけば、多くの場合問題ありません。

2.2　変数の宣言・代入・参照

変数をプログラムで使う場合、大きく「宣言」「代入」「参照」の三つの手順が必要になります。

変数の宣言

変数を使う場合、まずは名前をつける必要があります。これを「変数の宣言」と言います。変数の宣言は、以下のように、型名に続けて名前を記入します。

```
int number;   // number という名前の整数型変数の宣言
double value; // value という名前の実数型変数の宣言
char letter;  // letter という名前の文字型変数の宣言
int x, y;     // カンマで区切って同時に複数宣言することも可能
```

値の代入

変数に数字や文字などのデータを入れるには、代入演算子「=」を用います。ここで、「=」は数学の「等しい」という意味ではなく、「(右辺を左辺に) 代入」を意味することに注意して下さい。また、宣言された変数に初めて値を代入することを「**初期化**」と言います。初期化されていない変数を演算させたり、参照したりしようとすると、当然エラーが発生します。

```
1  // 変数の宣言
2  int x, y;
3  double value;
4  char letter;
5
6  // 値の代入 ( 初期化 )
7  x = 10;
8  y = -5;
9  value = 3.1415;
10 letter = 'P';
11
12 // 値の代入 ( 更新 )
13 x = y;
```

宣言した変数に対して、7〜10 行目でそれぞれ値を代入しています。文字型の変数 letter には、1 文字だけを格納することができます。1 文字（文字定数）は、上記ソースコードのようにシングルクォーテーション ‘’ で囲って表現します。また、13 行目では変数 x の値を y の値で上書きしています。「=」を等号と考えた場合、「x = y」は間違った等式になりますが、C 言語では「x に y の値を代入する」という意味になり、間違いではありません。この代入の前、「x = 10」「y = -5」でしたが、代入後は「x = -5」「y = -5」となります。

上のソースコードでは、「変数の宣言」と「値の代入 (初期化)」を独立して行っています。これらの操作は、以下のように同時に行うこともできます。覚えておきましょう。

```
1  // 変数の宣言と初期化
2  int x = 10, y = -5;
3  double value = 3.1415;
4  char letter = 'P';
```

値の参照

変数に入っているデータを出力するためには、第 1 章でも使った「printf」関数を用います。「printf」で変数を出力するには、以下の形で記述します。

```
printf(“フォーマット指定子”, 変数);
```

フォーマット指定子は「%」からはじまる書式で、実行時に変数に置き換えられます。フォーマット指定子としては、以下のようなものがあります。

型名	データの種類	指定子
int	整数値	%d
long	倍長整数値	%ld
float	実数値	%f
double	倍精度実数値	%lf
char	文字	%c

実際に値を参照するコードを書いて確認しましょう。

演習 2-1 変数の表示の練習（ex2-1.c）

以下のソースコードを参考に、年齢・身長・血液型を表示するプログラムを作成して下さい。

```c
#include <stdio.h>

int main(void) {
    // 変数の宣言と初期化
    int age = 29;
    double height = 175.8;
    char blood = 'A';

    // 変数の参照
    printf("   age = %d\n", age);
    printf("height = %lf\n", height);
    printf(" blood = %c\n", blood);

    return 0;
}
```

演習 2-1 実行例

```
$ ./a
    age = 29
height = 175.800000
 blood = A
```

変数表示の詳細設定

「printf」で変数を表示させる際、表示する桁数などを指定することができます。桁数の指定は、「%[全体の幅]d」や「%[全体の幅].[小数点以下の幅]」といった形で行います。また、「%」の直後に「0」を書くと「0 詰め」、「-」を書くと「左寄せ」、「+」を書くと「符号表示」になります。実際に以下のコード例を見てみましょう。

例 2-1

```c
#include <stdio.h>

int main(void) {
    // 変数の宣言と初期化
    int age = 29;
    double height = 175.8;
    char blood = 'A';

    // 変数の参照（幅 7 桁、小数以下 2 桁）
    printf("[%7d]\n", age);
    printf("[%7.2lf]\n", height);
    printf("[%7c]\n\n", blood);

    // 変数の参照（幅 7 桁、小数以下 2 桁、0 詰め）
    printf("[%07d]\n", age);
    printf("[%07.2lf]\n", height);
    printf("[%07c]\n\n", blood);

    // 変数の参照（幅 7 桁、小数以下 2 桁、左寄せ）
    printf("[%-7d]\n", age);
```

```
21      printf("[%-7.2lf]\n", height);
22      printf("[%-7c]\n\n", blood);
23
24      // 変数の参照（幅 7 桁、小数以下 2 桁、符号の表示）
25      printf("[%+7d]\n", age);
26      printf("[%+7.2lf]\n", height);
27      printf("[%7c]\n\n", blood);
28
29      return 0;
30  }
```

例 2-1 実行結果

```
$ ./a
[     29]
[ 175.80]
[      A]

[0000029]
[0175.80]
[000000A]

[29     ]
[175.80 ]
[A      ]

[    +29]
[+175.80]
[      A]
```

このように、変数表示の詳細を指定すれば、桁数が異なる変数も綺麗に揃えて出力することができます。

演習 2-2 変数を揃えて表示（ex2-2.c）

例 2-1 のソースコードを参考に、演習 2-1 のソースコードの結果を右揃えで出力して下さい。

演習 2-2 実行例

```
$ ./a
   age =    29
height = 175.8
 blood =     A
```

2.3　変数の命名規則

変数の名前の付け方には、「守らないといけないルール」と「守ったほうが良いルール」があります。それぞれを見ていきましょう。

守らないといけないルール

以下のルールは必ず守る必要のあるもので、守れていない場合はエラーが発生します。

- 半角のアルファベット、数字、アンダースコア（_）のみを用いる。
- 先頭は必ずアルファベットかアンダースコアを用いる（数字は不可）。
- 予約語は使用できない。

ここで予約語とは、C 言語ですでに意味を持っているワードのことで、「int」「double」「if」「switch」「for」「do」「while」「goto」「return」などがあります。

守ったほうが良いルール

以下に示すルールは、C 言語で変数名を付ける際のマナーのようなものです。必ず守る必要があるわけではありませんが、守ることで読みやすさや理解のしやすさが向上します。

- a, b, c など、意味のわかりにくい名前はつけない。
- アルファベットは全て小文字を用いる（定数の場合は全て大文字）。
- 複合語は単語の間にアンダースコアを入れる（スネークケース）。

名前の付け方には、「スネークケース」「キャメルケース」「パスカルケース」と呼ばれるものがあります。スネークケースは「snake_case」のように、全て小文字で単語の間にアンダースコアを入れる表記法、キャメルケースは「camelCase」のように、小文字で始まり単語の切れ目で大文字を用いる表記法、パスカルケースは「PascalCase」のように、単語の始ま

りを全て大文字にする表記法です。一般的に、C 言語の変数名にはスネークケースによる記法が用いられます。以上のことから、「利用不可な例」「悪い例」「良い例」をまとめて表にすると、次のようになります。

利用不可な例	悪い例	良い例
c-3	a, b, c	pos_x, pos_y, pos_z
!?#	a1_1, a1_2	position, velocity
(^-^)	l1, ll, l1I	num, value, count
2016_Oct	getdata	get_data
return	GETTEXTDATA	get_text_data

2.4　定数

「変数」はその名のとおり、プログラム中で値が変化する可能性があるデータです。一方、プログラム中に値が決して変化しない「定数」というデータを利用することもできます。定数は、通常「include 文」の下（main 関数の外）に、「#define 定数名 値」といった形で定義します。行の終わりを表すセミコロン「;」は付けません。また、定数は変数との区別と容易にするため、一般的に全て大文字で書きます。以下のコードを見て使用例を確認しましょう。

例 2-2

```c
#include <stdio.h>
#define RATE 101.338 // 為替レート

int main(void) {
    printf("現在のレートは 1 ドル = %lf 円です。\n", RATE);
    return 0;
}
```

例 2-2 実行結果

```
$ ./a
現在のレートは 1 ドル = 101.338000 円です
```

以上のように、定数の参照は変数と同様に行うことができます。ただし、変数とは違い、プログラムの途中で値を変更することはできません。

2.5　演算

C 言語での基本的な演算について、見ていきましょう。算術演算を行うための代表的な「算術演算子」を、以下に示します。

演算子	役割	例	意味
+	加算	z = x + y;	x に y を足した値を z に代入
-	減算	z = x - y;	x から y を引いた値を z に代入
*	乗算	z = x * y;	x に y を掛けた値を z に代入
/	除算	z = x / y;	x を y で割った値を z に代入
%	剰余算	z = x % y;	x を y で割った余りを z に代入

演習 2-3 算術演算子による整数の計算（ex2-3.c）

int 型の変数 x, y, z を用意し、x と y に適当な値を入れ、上の表に示した 5 つの算術演算の結果を表示して下さい。

演習 2-3 コード例

```c
#include <stdio.h>

int main(void) {
    int x = 18, y = 4;

    printf("x = %d, y = %d\n", x, y);
    printf("x + y = %d\n", x + y);
    printf("x - y = %d\n", x - y);
    printf("x * y = %d\n", x * y);
    printf("x / y = %d\n", x / y);
    printf("x %% y = %d\n", x % y);

    return 0;
}
```

※「%」はフォーマット指定子で使う記号であるため、
「%」のみを出力したい場合は「%%」と書きます。

演習 2-3 実行例

```
$ ./a
x = 18, y = 4
x + y = 22
x - y = 14
x * y = 72
x / y = 4
x % y = 2
```

ここで、「18 ÷ 4」の値が「4」になっていることに注意して下さい。このように、int 型（整数型）同士の演算結果は、小数点以下が切り捨てられ、必ず整数値になります。また、double 型（実数型）での計算を行ったとしても、int 型の変数に格納しようとした場合、同様に整数化されます。

演習 2-4 算術演算子による実数の計算（ex2-4.c）

double 型の変数 x, y, z を用意し、x と y に適当な値を入れ、「加算」「減算」「乗算」「除算」の結果を表示して下さい（実数値の「剰余算」は「%」演算子で行うことができません）。

演習 2-4 実行例

```
$ ./a
x = 18.00, y = 4.00
x + y = 22.00
x - y = 14.00
x * y = 72.00
x / y = 4.50
```

算術演算子 ＋ 代入演算子

算術演算子は、以下のように代入演算子と組み合わせることができます。

演算子	役割	例	意味
+=	加算代入	x += y;	x に y を足した値を x に代入
-=	減算代入	x -= y;	x から y を引いた値を x に代入
*=	乗算代入	x *= y;	x に y を掛けた値を x に代入
/=	除算代入	x /= y;	x を y で割った値を x に代入
%=	剰余算代入	x %= y;	x を y で割った余りを x に代入

これらの演算は「x＝x＋y」のように書いても同じ結果となります。しかし、加算代入演算子などを用いることで、コードを短く整理することができます。

インクリメント/デクリメント演算子

プログラムでは、変数を 1 ずつ増やしたり減らしたりすることがよくあります。そのようなときは、インクリメント演算子（++)、デクリメント演算子（--）を用いることができます。実際には、以下のように整数型の変数に繋げて用います。

```
1 int index = 0;
2
3 index++; // index に 1 を足す
4 index--; // index から 1 を引く
```

これら演算子は「++index」「--index」のように、変数の前に置くこともできます。この例のように単独で用いる場合は全く同じ結果になりますが、代入演算子「=」や比較演算子（第 3 章参照）と同時に用いると、異なる挙動を示すので注意が必要です。

```
1 int n, index = 0;
2
3 n = index++; // n に index を代入してから 1 を足す
4 n = ++index; // index に 1 を足してから n に代入する
```

インクリメント/デクリメント演算子は、通常の加算/減算処理よりも高速に動作します。そのため、何度も繰り返し 1 を加算したり減算したりする場合は、これら演算子を用いるようにしましょう。

2.7　キーボードからの入力

インタラクティブなプログラムを作成するには、キーボードなどからの入力を受け取る必要があります。データの入力には、「scanf」関数を用います。実際に、int 型の変数と double 型の変数を受け取る例を見てみましょう。

例 2-3

```c
#include <stdio.h>

int main(void) {
    int age;
    double height;

    printf("年齢を入力して下さい: ");
    scanf("%d", &age);

    printf("身長を入力して下さい: ");
    scanf("%lf", &height);

    printf("age = %d, height = %.2lf\n", age, height);

    return 0;
}
```

例 2-3 実行結果

```
$ ./a
年齢を入力して下さい: 29
身長を入力して下さい: 175.8
age = 29, height = 175.80
```

このように「scanf」では、読み取りたい変数の型のフォーマット指定子（「printf」のものと同じ）を記入します。そしてカンマで区切り、値を格納する変数に「&」記号を付けて記入します。「&」は変数のアドレス（住所）を表す記号で、値を格納する場所をコンピュータに教えてあげています（詳細は第 6 章「配列とポインタ」にて）。また、「scanf(“%d %lf”, &age, &height);」と書いて、複数の変数を同時に読み取ることも可能です。ただし、入力時には数値をスペースで区切る必要があります。

演習 2-5 データ入力の練習（ex2-5.c）

「scanf」を用いて生まれ年と現在の年を西暦で受け取り、年齢を出力するプログラムを作成して下さい。

演習 2-5 実行例

```
$ ./a
生まれ年（西暦）を入力して下さい：1987
現在の年（西暦）を入力して下さい：2016
2016 年、あなたは今年 29 歳です。
```

2.6　課題

課題 2-1

int 型、double 型、char 型の変数に対し、「整数値」「実数値」「文字」をそれぞれ代入するとどのようになるか、9 通りの結果をまとめて下さい。また、char 型の変数をフォーマット指定子「%d」で表示をし、その結果について考察をして下さい。

課題 2-2

7! と17! の値を計算して表示して下さい。また、結果について考察をして下さい。

（「!」は階乗を意味します。例えば、$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ となります。）

課題 2-3

円錐の底面の半径と高さを入力すると、体積を計算して表示するプログラムを作成して下さい。円周率は定数として定義しましょう。また、真値との誤差について、考察・検討をして下さい。