

■ 宣言文、データ型

□ 宣言文とデータの型

変数を利用するためにはプログラムのはじめに宣言文が必要です。

```
int i;  
i=1234;
```

といった形です。変数やデータ（の値）には「型（かた）」と呼ぶ概念があり、この宣言文では変数を型と共に宣言します。

「int i;」は、「int 型（がた）」の変数 i を宣言する、という意味です。

変数はその型に適合する値しか扱いません。int 型は整数を扱うための型で、int 型変数 i には整数のみを代入することができます。

```
#include <stdio.h>  
  
/*  
   型を確認する 473088 榎田裕一郎  
*/  
  
int main() {  
    char c;  
    int i;  
    float f;  
    double d;  
  
    c = 'x';  
    i = 1234;  
    f = 3.14159f;  
    d = 1.0e-5;  
  
    printf("char type : %c\n", c);  
    printf("int type : %d\n", i);  
    printf("float type : %f\n", f);  
    printf("double type : %e\n", d);  
  
    return 0;  
}
```

同じデータ型の変数なら、「int i, j, k;」と、カンマで並べて一度に宣言できます。

代表的な型を以下に示します。

char : 文字(character)を扱うための型。'x', 'a' など。

int : 整数(integer)を扱うための型。123, 0, -3 など。

float, double: 実数を扱うための型。123.456, 0.0001, 0.0, 1024.0 など。

float と double の違いは精度（有効桁数）です。float の倍の有効桁数を double は持っています。123456.78901234 など。float を単精度浮動小数点型、double を倍精度浮動小数点型と呼ぶ場合があります。

変数の命名規則については補足資料を参照。

□ 定数における型の明記

整数と実数の型の区別は小数点で行います。123 は整数、123.456 と小数点をつけた数値は実数とみなされます。123 と 123.0 は数値としては同じですが、型は異なる、というわけです。

123.456 は実数でも double 型とみなされます。float 型と明記したい場合は 123.456f と最後に f をつけて表記します。（サンプルプログラムの f = 3.14159f; を参照。）
実数はまた 1.2e-5; というように表記できます。（ 1.2×10^{-5} つまり 0.000012）

● 課題 1.

上のサンプルプログラムを入力し、実行して結果がどうなるか調べてください。

□ 型変換

int 型変数は整数しか扱えません。そこで int 型変数に実数を代入すると、自動的に型変換が行われ、小数部が切り落とされて整数部だけが代入されます。

つまり `int i; i=123.456;` なら、i には少数以下が切り落とされた 123 が代入されます。

実数変数に整数を代入した場合は、単純に実数化された値が入ります。 `float a; a=123;` なら、123 が実数化されて `123.0f` となって a に代入されます。

計算式に整数、実数の値や変数が混在していた場合は、精度の高い方に合わせて型変換が行われてから計算が行われます。 `float a; a=1.5f * 3;` なら、`1.5f * 3` は `1.5f * 3.0f` として計算され、`4.5f` が a に代入されます。

注意が必要なのは / (割り算) で、整数と整数の割り算は整数となって余りは捨てられますが、実数と実数、または実数と整数の割り算では可能な限りの精度で小数点以下まで求められます。

つまり `10/4` は 2 ですが、`10.0/4.0` は 2.5 です。変数を用いた計算でも同じことで、

```
int i, k; float a, b;  
i=10; k=4; a=10.0; b=4.0;
```

の場合、

`i/k` は 2 ですが、`a/b` は 2.5 です。

`i/b` や `a/k` のように実数と整数を混在させた場合は、整数側が実数化されて計算され、結果はともに 2.5 になります。

■ 型に合わせた変換文字列

`printf()` で変数の値を表示させるためには `%d` といった変換文字列を使っていました。変換文字列は `%d` だけではなく、型と表示桁数の制御を指示する事ができます。(補足資料も参照)

例 :

`%5d` 5 桁で整数型の値を表示

`%6.2f` 全体で 6 桁、ただし小数点以下 2 桁で実数型の値を表示 (123.40 というように)

● 課題 2

右のサンプルプログラムを入力し、実行して結果がどうなるか調べてください。何桁目で桁あふれするか、値がどうおかしくなるか、調べて見ると良いでしょう。

```
int main() {  
    float f;  
    f = 3.14159f;  
    printf(" 指定無し           : %f\n", f);  
    printf("           5.2 : %5.2f\n", f);  
    printf("           5.3 : %5.3f\n", f);  
    printf("           6.3 : %6.3f\n", f);  
    printf(" 全長無し   .3 : %.3f\n", f);  
    printf(" 溢れる場合 5.3 : %5.3f\n", 123.456);  
    return 0;  
}
```

時間に余裕のある受講生は、次週のグラフィクスプログラムを試しに入力して走らせてみて下さい。プログラムは教材 Web の資料の項にあります。

■ 補足資料

□ 変数名と予約語

変数は宣言時に名前がつけられますが、変数名として利用可能なのは以下のようなものです。

- ・最初の一文字は英字（大文字、小文字のいずれでも）ではじまること。
- ・二文字目以降は英字または数字が使える。
- ・変数名は長くても 31 文字まで。
- ・変数名も大文字と小文字は区別される。（C の変数は伝統的に小文字で表記される。）
- ・予約語以外の名前であればならない。

予約語とは `int` や `float`、`return` といった、C 言語の文法上、既に使われている単語のことです。以下に予約語の一覧を示します。

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

なお、C 言語では「`_`」（下線、アンダーライン）は英字とされています。長い変数名をわかりやすくする場合などに使われます。（`leftbutton` より `left_button` の方が読みやすい。）

注意：予約語ではないが衝突を避けたい名前

`printf()` などは C 言語の予約語ではなく「一般的な関数」として標準的に UNIX システムに用意されているものです。この名前と衝突するのも良くありませんので、注意しましょう。こうした標準的なライブラリ関数の名前は多すぎてここには出しません。

□ データの型に合わせた変換文字列

サンプルプログラムの `printf()` を見てください。文字は `%c`、整数は `%d`、実数は `%f` または `%e` と、変換文字列が使われています。

```
printf("char type : %c\n", c);
printf("int type : %d\n", i);
printf("float type : %f\n", f);
printf("double type : %e\n", d);
```

`%10d` のように、`%` と変換文字の間に桁数の指定などができます。

代表的な変換文字列

| | 意味 | 使用例 | その結果 |
|-----------------|-------|---|--|
| <code>%c</code> | 文字を表示 | <code>printf("[%c]\n", 'a');</code> | <code>[a]</code> 文字がそのまま表示される。 |
| <code>%d</code> | 整数を表示 | <code>printf("[%d]\n", 10);</code> <code>printf("[%5d]\n", 10);</code> <code>printf("[%05d]\n", 10);</code> | <code>[10]</code> 桁数不定 <code>[10]</code> 5桁で表示。不足分は空白。 <code>[00010]</code> 5桁。不足はゼロで埋める。 |
| <code>%f</code> | 実数を表示 | <code>printf("[%f]\n", 12.345);</code> <code>printf("[%9.5f]\n", 12.345);</code> | <code>[12.345000]</code> 桁数不定 <code>[12.34500]</code> 小数点含めて全体が 9桁、小数以下が 5桁。 |
| <code>%e</code> | 実数を表示 | <code>printf("[%e]\n", 12.345);</code> | <code>[1.234500e+01]</code> 浮動小数点で表示 |