プログラミング演習 A 教材 (#4)

■ 宣言文、データ型

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。ファイル名は任意で構いません。

```
#include <stdio.h>

/*
  型を確認する 473088 榎田裕一郎
*/

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;
}
```

□ 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(*1)。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。

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(*2)。

注意：文字と文字列
C では文字と文字列を全く違うものとして扱っています。文字とは長さが 1 に固定されており、`'x', 'a'` などとシングルクォーテーションで囲んで表現します。文字列とは長さが不定で、`"sample", "hey, Jude"` のように引用符（ダブルクォーテーション）で囲まれています。
つまり `"x"` は「長さが 1 の文字列」であり、`'x'` の「文字 x 」とは異なるものです。
C 言語には文字型はありますが、文字列型は（興味深いことに）用意されておらず、文字の配列として文字列を表現します。配列についての説明は「プログラミング B」で行います。

*1 宣言文無しに変数を使おうとした場合はコンパイルエラーになります。エラーメッセージは：
`sample.c:3: `i' undeclared (first use in this function)`
*2 単精度、倍精度の意味や、具体的な桁数、浮動小数点の意味などについては「＊＊＊」クラスで説明します。

□ 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` ）

□ 型変換、キャスト

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。

これらは暗黙の型変換と呼ばれますが、明示的に指示して行うこともできます。

もし、`i / k` の計算を実数化して行い、2.5 という結果を実数変数に代入したい場合は、
`float x; x = (float)i / (float)k;` と書きます。

こうした、値の直前にカッコで囲んで型名を書く「キャスト」と呼ばれる方法で、値の型変換を明示的に指示できます。

キャストの有効範囲がどこまでか不安になるような場合があり得ます。例えば、
`x = (int) a / k * 100;` のようなケースは、
`x = (int)(a) / k * 100;` として a だけキャストされるのか、
`x = (int)(a / k * 100);` のように、全体の計算結果に最後に一度だけキャストされるのか、不安に思うかも知れません。そうしたときは、なるべくはっきりキャストの範囲が明確にわかるよう、上記のようにカッコをうまく使って記述すると良いでしょう。

□ 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なければならない。

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。

```
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
```

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。
変数名を決める際に、同じ名前のライブラリ関数があるかもしれない、と不安になった場合は man コマンドで確認すると良いでしょう。たとえば man 3 printf とすると printf( ) 関数の詳細なマニュアルが表示されるでしょう。

□ 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、変換文字列が使い分けられています。

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