プログラミング演習 A 教材 (#3)

■ 計算式、変数と代入

□ 簡単な計算を含むプログラム

Emacs を起動して、下のプログラムを入力、コンパイル、実行してください。時、分、秒をもとに、0 時から数えた秒数を計算して表示するものです。
サンプルでは 10 時 32 分 45 秒としていますが、時間は適当に現在時刻を入れて下さい。ファイル名は任意の名前で結構ですが、例えば comp.c とでもしておいてください。

```
#include <stdio.h>

/*
  秒数を計算する 473088 榎田裕一郎
*/

main() {

  printf("今は %d 時 %d 分 %d 秒です\n", 10, 32, 45);
  printf("全部で %d 秒めですね\n", 10 * 3600 + 32 * 60 + 45);

  return 0;
}
```

押さえて欲しいポイント：
・/* と */ で囲まれた範囲はコメントです。
（コンパイル時には無視されるので、プログラム以外の覚え書きなどを書いておくのに便利。）
・C プログラムの中では数値や計算式が書ける。
・数値を printf で表示させるためには %d といった変換文字列を使う。
・return でプログラムの実行を終了する。返り値（ステイタス）は 0 （正常終了を意味する）。

□ 計算式、演算子、優先順位

C プログラムのなかでは各種の計算が記述できます。四則演算の表記法は以下の通りです。
+, * といった記号を演算子と呼んでいます。

演算子	意味	表記例と計算結果
+	加算（＋）	20 + 6 （結果は 26 になる）
-	減算（－）	20 - 6 （結果 は 14 になる）
*	乗算（×）	20 * 6 （結果 は 120 になる）
/	除算（÷）	20 / 6 （結果 は 3 になる）
%	剰余（割った余り）	20 % 6 （結果 は 2 になる）

整数どうしの割り算 (/) の結果は（小数点以下を切り捨てて）整数となります。
例えば 6 / 20 は 0 です。

通常の計算式と同じく、演算子には優先順位があります。*, /, % が +, - より高い優先順位で処理されます。また、カッコを使った演算順序の制御もできます。また、例では読みやすくするために適度に空白を入れていますが、詰めて書くことも可能です。
つまり以下の計算式はどれも同一の結果を返します。

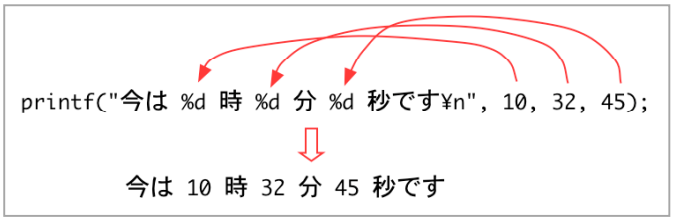
```
10 * 3600 + 32 * 60 + 45
( 10 * 3600 ) + ( 32 * 60 ) + 45
( ( 10 * 3600 ) + ( 32 * 60 ) + 45 )
( ( ( 10 ) * 3600 ) + ( 32 * 60 ) ) + 45
(10*3600)+(32*60)+45
```

□ printf　の変換文字列

printf( ) のカッコのなかにはカンマで区切られた幾つかの項目があります。これらは引数（実引数または argument (*1) ）と呼ばれ、そこには数値 (10, 32 など)、文字列 ("Hello" など)、計算式 (10 * 3600)などが書けます。

最初の引数には文字列を与えます。printf はこの文字列の記述に従って出力結果を整形します。たとえば「今は %d 時」の %d は「ここに 10 進数(decimal) の数値をあてはめる」という指定で、第二の引数である 10 が対応します。こういった % に続く書式（フォーマット）の指定を行う表現のことを「変換文字列」と呼んでいます。

続く二番目の %d には第三引数である 32 が、三番目の %d には第四引数の 45 が対応し、最終的には「今は 10 時 32 分 45 秒です」と整形されて出力（画面に表示）されます。



*1 仮引数(parameter)、実引数(argument) の詳しい用語の定義などについては「プログラミング B」クラスで説明します。

注意：数値と文字列のなかの数字
C 言語のなかでは文字列としての数字と、値として扱われる数値はまったく別のものです。
1 + 2 は数値による計算式で、最終的に 3 という数値になって処理されます。
"1 + 2" は数字を含む文字列であり、数値になることはありません。
つまり、

```
printf(“12+34\n”);
printf(“%d\n”, 12+34);
```

は全く違う結果を画面上に表示します。その理由が納得できましたか？

□ 変数と代入

C 言語（とそれ以外の多くのプログラミング言語）には変数というものがあります。以下にサンプルを示します。結果として「合計は 3 です」と表示されることが想像つくでしょう。

```
int a, b, c;

a = 1;
b = 2;
c = a + b;
printf("合計は %d です\n", c);
```

変数とは値を入れる容器のようなもので、C 言語の場合は中に一つだけ値を入れることが出来ます(*1)。この値を入れる作業を代入と呼び、値を入れるには以下のような代入文を用います。

a = 1;

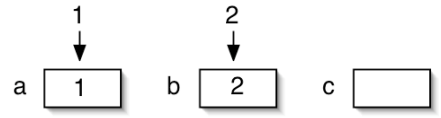
イコールの右辺の値（または計算結果）が左辺の変数に書き込まれます。左辺には一つの変数しか書けません。

ざっと動きを図解すると以下のようになります。

STEP 1.

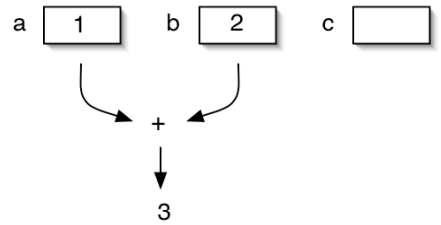
int a, b, c で三つの変数を用意。
C 言語では変数をはじめに宣言します。

STEP 2.



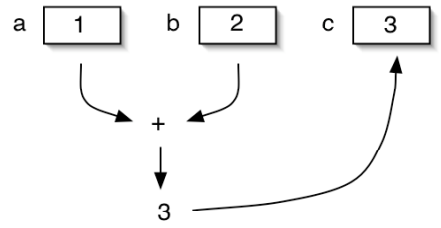
= によって変数 a と b に値を入れる（代入）

STEP 3.



a + b を計算
（ a の中身と b の中身を足す）

STEP 4.



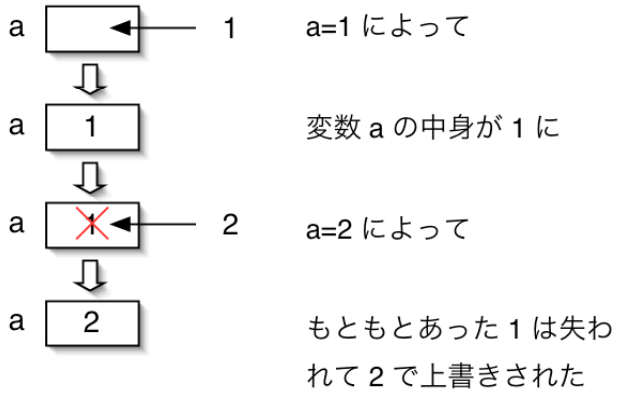
その結果（値）を変数 c に入れる（代入）

*1 配列変数、という手法を使って少し違うこともできます。「プログラミング B」クラスで説明します。

変数には一つの値しか入れられませんから、変数に二度目に値を代入すると以前の値は失われ、それに置き換わって新しい値が残ります。

```
a = 1;
a = 2;
printf("%d\n", a);
```

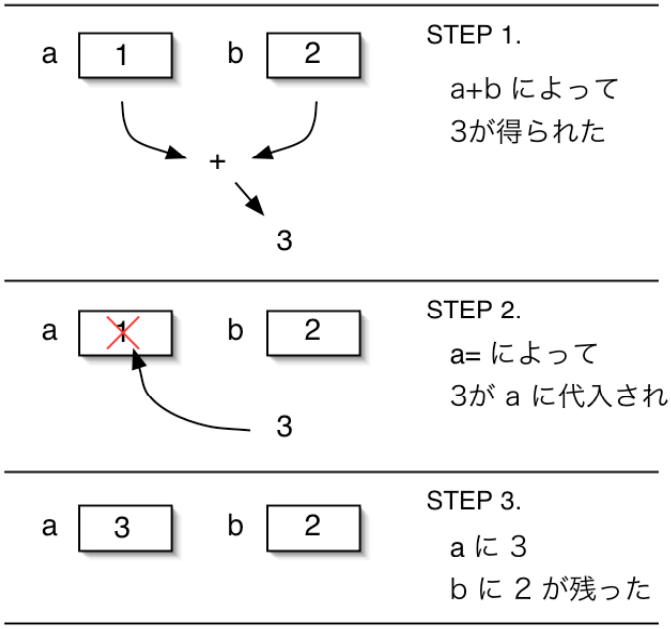
printf( ) の結果は 2 になるでしょう。



計算結果を自分自身に代入することもできます。以下のサンプルの a = a + b に注目して下さい。

```
a = 1;
b = 2;
a = a + b;
printf("%d\n", a);
```

printf( ) の結果は 3 になるでしょう。



重要なこと：
代入文では、右辺の計算が完全に終わってから左辺の変数に代入されます。
右図を見ても分かるように、プログラムには順序と時間の経過が重要なのです。

課題：
冒頭の秒数を計算するプログラムを、変数を使って書き直せ。具体的には `printf()` の記述を以下のように変更し、その前に各変数の値を設定・計算する代入文を加える。

```
printf("今は %d 時 %d 分 %d 秒です\n", hour, min, sec);
printf("全部で %d 秒めですね\n", total);
```