Practice A を解けるようになるために

『PracticeA - はじめてのあっとこーだー（Welcome to AtCoder）』を見てみる.

この問題が解けるようになるには･･･？

- 整数 a, b, c と文字列 s を**入力** →入力処理,入れ物
- a + b + c を**計算** → 足し算
- a + b + c と s を**出力** → 出力処理

つまり, 入力する処理, 入力したものをいれるもの, 四則演算, 画面に表示する処理を学ぼう.

画面に文字を表示する

まずは画面上に

```
Hello, World!
Good Bye, World.
```

と表示するプログラムを書いてみよう.

Sample1_1.c

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    printf("Good Bye, World.\n");
    return 0;
}
```

実際に「Hello, World!」という表示を行っている部分は,

```
printf("Hello, World!¥n")
```

の部分だけ.

現時点では, 以下の点について理解していればよい.

- 処理はまず　int main()の後の中括弧の中から始まる.上から順に命令が実行される.
- 命令の後には必ずセミコロン「;」を入れる.
- printf は関数である. 数学における「関数」とは意味が微妙に違う.今のところは「括弧の中に何かを入れると, いろいろ処理をしてくれるもの」という理解で十分. 詳しい話は「関数」の項で学ぶ.
- printf(“〇〇〇”)と入力すれば, 〇〇〇を表示してくれる.
- printf 中にある『\n』は, 改行を表す文字.
- printf を呼び出すためには, #include <stdio.h>を書かなければならない.
- return 0 でプログラムが終了する. この文の詳しい意味については「関数」の項で学ぶ.

数字を間接的に表示する

37 + 24 の結果を表示するプログラムを作ってみよう.直接 61 を表示するプログラムを作ってみてもいいが, 今回は実際にコンピュータ内部で計算させてみる.

Sample1_2.c	

```c
#include <stdio.h>

int main() {
    printf("37 + 24 = %d\n", 37 + 24);
    return 0;
}
```

実行結果は以下のようになる.

```
37 + 24 = 61
```

printf(“37 + 24 = %d”, 37 + 24);

とは,「『37 + 24 = %d』を表示しろ.ただし, %d は 37 + 24 を計算した値に読み替えてくれ.」という意味.

%d というのは,「カンマで区切られた先にある値を 10 進整数表記で表示せよ」という意味.10 進整数以外の値の表示は%d ではない.

例えば, 浮動小数点数(現時点ではただの小数だと思っておけばよい)を表示したいなら%f, 文字を表示したいなら%c, 文字列を表示したいなら%s と書く.また, 複数の値を一つの printf で表示したいなら, 次のように書く.

```
printf("%d %c %d %f %s", 123, 'A', 128, 3.14, "ABC");
```

%〇の順序が, 後のカンマで区切られた値の順序に対応している.

[補足]

文字は「一文字」を表し, これを表現するためにはシングルクォーテーションでくくる.文字列は「文字の集まり」を表し, これを表現するためにはダブルクォーテーションでくくる.

変数の宣言と代入

プログラミングの世界には「変数」という言葉がある.

数学で習った「変数」とは微妙に意味が違う.

プログラミングにおける変数とは, 簡単に言うと

何か値を入れておくための箱

である.箱の中に必要なものを入れておけば, それを後で計算に用いたり, 書き換えたりできる.

箱には(基本的には)何でも入れていいわけではない.入れるものに適した箱が必要である.また, 箱に入れられるものの大きさには限度がある.

変数においても同じことが言える.箱の種類のことを「型」と呼ぶ.変数には整数型や文字型, 浮動小数点数(小数)型などの変数がある.それぞれの箱には決まった種類の値しか(基本的には)入れられない.また, 型によって入れられる値の大きさが決まっている.ただし, 変数一つにつき一つの値しか入れられないことに注意.

それを踏まえて, 変数を使ったプログラムのコードを見てみよう.

Sample1_3.c

```
#include <stdio.h>

int main() {
    int val1, val2;
    val1 = 10;
    val2 = val1 + 35;

    printf("val1 is %d\n", val1);
    printf("val2 is %d\n", val2);

    return 0;
}
```

実行すると以下のような結果となる.

```
val1 is 10
val2 is 45
```

{}内の処理について順に説明する.

int val1, val2;

これは「俺は int(整数)型の変数 val1 と val2 を宣言する！」ということを意味している.以下, 詳しく説明する.

変数を作る処理を「変数の宣言」と呼び, 次の書式で書く.

型 変数名;

型には様々な種類があるが, 今のところは

int, char, double

の三つを知っておけばよい.それぞれ整数, 文字, 浮動小数点数を表す.

変数名は,「他の変数と名前が重複しない」,「予約語(C 言語が持っている特別な単語)でない」限りは, どんな名前でもよい.

カンマで区切ると, 同じ型の変数を複数宣言できる.

次に,

```
val1 = 10;
```

によって,「val1 に 10 を代入する」という処理を行う.

```
変数名 = 値;
```

という文によって, 作った変数の中に値を入れることができる.これを「変数に値を代入する」という.

※「文ってなんだよ.説明されていないぞ.」と思った人は鋭い.しかし少し細かい話になるので説明しない.使い方からなんとなく理解してくれ.

同様に,

```
val2 = val1 + 35;
```

という文は「val2 に(val1 + 35)を代入する」ことを意味する.プログラムを実行したときに, 右辺は

```
val1 + 35 = 10 + 35 = 45
```

と読み替えられ,「val2 に 45 を代入する」という意味になる.

変数への代入を行った後は, それらを printf で表示している.

```
printf(“val1 is %d\n”, val1);
```

前の項で説明した通り, %d は val1 の値に置き換えられる.プログラムを実行したときに, val1 の部分はその中身 10 に読み替えられ,

```
printf(“val1 is %d\n”, 10);
```

と同じ意味になる.

[補足]

上の説明で「読み替えられ」という表現を用いたが, 厳密にはこの読み替え作業のことを「評価」と呼ぶ.

例: val1 = 10 のとき, (val1 + 35)を評価すると 45 となる.

変数の初期化

初期化とは, 変数の宣言と同時に値を入れることである.初期化は以下のように書く.

型 変数名 = 値;

変数の宣言と代入が入り混じったような書き方である.前項のプログラムを変数の初期化で書き換えると以下のようになる.

Sample1_4.c

```c
#include <stdio.h>

int main() {
    int val1 = 10;
    int val2 = val1 + 35;

    printf("val1 is %d\n", val1);
    printf("val2 is %d\n", val2);

    return 0;
}
```

実行結果は前項と同じ.

ちなみに, もし初期化されていない変数を表示すると, どんな値が出るかわからない(この値を「不定値」と呼ぶ).何が入っているかわからないので, 初期化または代入を行っていない変数を使ってはいけない.

演算

いままでのプログラムでも何気なく足し算を使っていたが, 今回はそれ以外の演算も見てみよう.

Sample1_5a.c

```c
#include <stdio.h>

int main() {
    int a = 9;
    int b = 4;

    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n", a - b);
    printf("a * b = %d\n", a * b);
    printf("a / b = %d\n", a / b);
    printf("a % b = %d\n", a % b);

    return 0;
}
```

実行結果は以下のようになる.

```
a + b = 13
a - b = 5
a * b = 36
a / b = 2
a % b = 5
```

+, -, *, /はそれぞれ加減乗除を表す.ここでの%は, a%b という形で「a を b で割った余り」を表す.

計算結果がどんな型になるのかは, 計算に用いた値の型による.整数同士の計算なら計算は整数になる(除算のとき, 小数点以下は切り捨てられる).

数学でもいえることだが, 一般に+や-などの, 演算を表す記号のことを「演算子」という.演算の対象となる変数や値は「オペランド」と呼ばれる.

演算子は上で紹介したもののほかにもいくつかある.ここではすべては紹介しないが, いくつかをソースコードとともに見せよう.

Sample1_5b.c	

```c
#include <stdio.h>

int main() {
    int x = 0;
    printf("x: %d\n", x);

    x++;
    printf("x: %d\n", x);

    x += 5;
    printf("x: %d\n", x);

    x--;
    printf("x: %d\n", x);

    return 0;
}
```

++はインクリメント演算子という.単に値を 1 増やす.同様に, --をデクリメント演算子という.これは値を 1 減らす.x += 5 は複合代入演算子の一種.その名のとおり代入と演算を一つにまとめたものである.つまり,

x = x + 5;

を省略した形.結局これは「今の x の値に 5 を加えよ」と述べている.なぜだろうか.

代入のイコールはは決して数学におけるイコールとは異なることに注意しよう(右辺と左辺が等しいわけではなく, 単に「代入」するための演算子と考える).

『変数の宣言と代入』の項でも言った「読み替え」の話を思い出そう.

もし x = 1 ならば, 右辺の x は 1 と読み替えられ,

x = 1 + 5

になる.よってこれは「今の x の値を 5 増やせ」と同じ意味になる.

同じような形として,

x -= a;

x *= a;

x /= a;

x %= a;

という複合代入演算子も同様に考えられる.実はインクリメント演算子やデクリメント演算子も複合代入演算子の一種で, それぞれ,

x = x + 1

x = x – 1

と同じ意味である.

キーボードから入力

前項のプログラムを改良して,「キーボードから入力した 2 つの整数に対して四則演算をした結果を表示する」プログラムを作る.

Sample1_6.c	

```c
#include <stdio.h>

int main() {
    int a, b;

    printf("input:");
    scanf("%d %d", &a, &b);

    printf("a + b = %d\n", a + b);
    printf("a - b = %d\n ", a - b);
    printf("a * b = %d\n ", a * b);
    printf("a / b = %d\n ", a / b);
```

```
    return 0;
}
```

実行結果は以下のようになる(*27 3* は入力例).

```
input: 27 3
a + b = 30
a - b = 24
a * b = 81
a / b = 9
a % b = 0
```

実は scanf について, 現時点ですべてを理解することができない.とにかく, キーボードから入力した整数値を変数に格納するためには,

scanf(“%d”, &変数名);

と書くことを覚えておこう.printf と似ているが, %d は入力した数を整数値として変数に格納することを表す.カンマ後に格納先の変数を指定する.ただし, その変数の先頭に&をつけなくてはならない(なぜつけなくてはならないのかを説明するには「ポインタ」を学ぶ必要がある).

2 つの入力をスペース区切りで入力するには, 次のように書く.

scanf(“%d %d”, &変数名, &変数名);

これで Practice A は解ける？

実は解けない.まだ「文字列をキーボードから入力」することについての説明がしていないから.

文字列は「配列」と密接な関係がある.配列について説明しなくてはならない.というわけで配列について説明する.

配列

変数ひとつにつき一つの値しか入れられないのだった。いくつかの変数をまとめて扱いたいときがある。それが配列の出番だ。

配列とは

　複数の箱が一直線上に並んだもの

と理解しておけばよい。それぞれの箱には番号がかかれている。

演習問題

1. Practice A の劣化版を解いてみよう.
   「整数 a, b, c が与えられたとき, a + b + c を表示せよ.」

   入力:

   入力は以下の形式で行われる.

   ```
   A
   b c
   ```

   1 行目は, 整数 a (1≦a≦1,000) が与えられる.

   2 行目は, 整数 b,c (1≦b,c≦1,000) が与えられる.

   出力:

   a + b + c を出力せよ.ただし, 出力の末尾には改行をいれること.

2. 好きな数字を入力し、それを表示するプログラムを作成せよ。

   入力:

   入力は以下の形式で行われる。

   ```
   K
   ```

   ただし k は整数であり、$-2^{31} < k < 2^{31} - 1$である。

   出力:

「I love (好きな数字).」と表示せよ。ただし(好きな数字)には入力した値 k を入れること。出力の末尾には改行を入れよ。

a.