初心者のための C言語講座

#3: 変数

Column）%が出力できない!!

　コラムから始まる講座資料とは。こんにちは、ZAKIです。今回の講座は、前回学んだことの補足から行いたいと思います。

　早速ですが問題です。

　実行結果が「%」となるようなプログラムを作成せよ。

▲ こんな実行結果

Column）%が出力できない!!

右上のようなプログラムを想像した方が大多数ではないでしょうか。

おめでとうございます。不正解です。

そうなんです。「%」という文字はそのままでは出力できません。というのも「%」は、「%d」に代表されるような変換指定文字の目印としての役割を担っているからです。要は、「%」は特別な意味を持つ文字なんですね。

したがって、「%」と出力したいときには、printf()内では「%%」と入力しなければなりません。

```
#include <stdio.h>

int main(void){
    printf("%¥n");
    return 0;
}
```

▲ これはダメ

```
#include <stdio.h>

int main(void){
    printf("%%¥n");
    return 0;
}
```

▲ これはOK

Column）%が出力できない!!

　同様の理由で、普通にprintf()内に書いても出力できない文字は、他にもあります。

　Figure3.1にそれらの文字をまとめておいたので、1度目を通しておいてください。

printf()内での表記	出力
%%	%
¥¥	¥
¥'	'
¥"	"

▲ Figure3.1　特殊な表記をする文字

#3-1）変数の型

はい、それでは本編突入です。今回は、前回もちょっと登場した「変数」について学んでいきましょう。

変数の宣言については、前回もやりましたね。「これからこういう名前の変数を使うから、宜しくね!!」と言った指示のことでした。右のプログラムの赤字部分がそれに当たります。

複数の変数を宣言するときは、「int = x, y, z」のように一括してやっても、右のように分けて宣言しても問題ありません。また、変数名に使用できる文字はA～Z、a～z、0～9、_(アンダーバー)に限られます。ただし、変数名の先頭を0～9の数字にするのは×です。

```
#include <stdio.h>

int main(void){
    int x;
    int y;
    int z;
    x = 15;    y = 30;    z = 45;
    printf("x = %d¥n", x);
    printf("y = %d¥n", y);
    printf("z = %d¥n", z);
    return 0;
}
```

▲ Program3.1　number.c

#3-1）変数の型

さて、ここで宣言されたx、y、zの3つの変数の「型」は何でしょうか？そうです。int型でしたね。intというのは、英語のinteger(整数)という単語が由来になっています。

その名の通り、int型とは整数型のことで、int型の変数には整数しか入れることが出来ません。このままでは、小数が扱えないので、困ってしまいます。

でも安心してください。ちゃんと小数を扱える変数の型も存在します。それでは、新しい変数の型を学んでいきましょう。

```c
#include <stdio.h>

int main(void){
    int x;
    int y;
    int z;
    x = 15;    y = 30;    z = 45;
    printf("x = %d¥n", x);
    printf("y = %d¥n", y);
    printf("z = %d¥n", z);
    return 0;
}
```

▲ Program3.1　number.c

#3-1）変数の型

　とりあえず、早急に覚えて欲しいものだけをFigure3.2にまとめました。（本当はもっとたくさんありますよ）

　えっ、分かりづらいから3行で説明しろ？　しょうがないですね。

整数使いたかったらint!!
小数使いたかったらdouble!!
大体これでイケる!!

　floatはしばらく脇に置いておくということで……。

型名	意味
int	整数型
float	実数型(単精度)
double	実数型(倍精度)

▲ Figure3.2　超基本的な変数の型

#3-2）倍精度実数型 double

　では、新しく覚えた「double」を使って、少し遊んでみましょう。

　右のProgram3.2を見てください。xおよびyという変数がdouble型で宣言されています。宣言の後には、それらに小数が代入されていますね。

　最後にそれらをprintf()で出力する訳ですが、1点注意があります。小数をprintf()内で出力するときの変換指定文字は「%f」です。ここを誤って「%d」としてしまうと、正しくない結果が出てきてしまいます。

```
#include <stdio.h>

int main(void){
    double x;
    double y;
    x = 1.5;    y = 2.5;
    printf("x = %f¥n", x);
    printf("y = %f¥n", y);
    return 0;
}
```

▲ Program3.2　number2.c

#3-2）倍精度実数型 double

実行結果は右下のResult3.2のようになります。変換指定文字を「%f」にした場合、出力結果は通常小数第6位まで表示されるようになっています。

小数第1位まで表示したいときには、変換指定文字を「%.1f」に、小数第2位まで表示したいときには、変換指定文字を「%.2f」に……という感じで表示の仕方は変えることが可能です。

```
#include <stdio.h>

int main(void){
    double x;
    double y;
    x = 1.5;    y = 2.5;
    printf("x = %f¥n", x);
    printf("y = %f¥n", y);
    return 0;
}
```

▲ Program3.2　number2.c

```
x = 1.500000
y = 2.500000
```

▲ Result3.2 number2.cの実行結果

Column）intとdoubleの落とし穴

コラム第2弾です。

問題です。右のProgram3.3の実行結果はどうなるでしょうか？

ちなみに、「*」は乗算(掛け算)、「/」は除算(割り算)の演算子です。

```
#include <stdio.h>

int main(void){
    int x, y, z;
    double w;
    x = 3;    y = 5;    z = x*y;    w = x/y;
    printf("x*y = %d¥n", z);
    printf("x/y = %.1f¥n", w);
    return 0;
}
```

▲ Program3.3　number3.c

Column）intとdoubleの落とし穴

結論から言ってしまうと、右下のResult3.3のようになります。

掛け算の方は、12×5=60なので問題ないでしょう。ただ割り算の方は、12÷5=2.0と意味の分からないことになってますね。このコンピュータ、ぶっ壊れてるんじゃないでしょうか。

もちろんコンピュータが壊れている訳ではありません。実はint同士の計算結果はintで出力されるような仕組みになっているのです。したがって、12/5の計算結果は2.4ではなく、それの小数部分を切り捨てた2となり、それがそのままdouble型変数wに代入されてしまったというお話です。

「int/int」の形なので、
計算結果もintに!!

```c
#include <stdio.h>

int main(void){
    int x, y, z;
    double w;
    x = 12;   y = 5;   z = x*y;   w = x/y;
    printf("x*y = %d¥n", z);
    printf("x/y = %.1f¥n", w);
    return 0;
}
```

▲ Program3.3　number3.c

```
x*y = 60
x/y = 2.0
```

▲ Result3.3 number3.cの実行結果

Column）intとdoubleの落とし穴

では、このような事態を回避する方法を教えます。

まず、変数x、yのいずれか、もしくは両方をdouble型で宣言する方法です。

こうすることで、件の計算部分は「int/double」または「double/int」または「double/double」の形をとるので、めでたく計算結果の形はdoubleになります。

いずれか一方でもdoubleであれば、それに引っ張られて計算結果もdoubleになることを覚えておいてください。これは除算に限った話ではなく、加算、減算、乗算などの他の演算のときにも言えることです。

int/double = double

```
#include <stdio.h>

int main(void){
    int x, z;
    double y, w;
    x = 12;    y = 5;    z = x*y;    w = x/y;
    printf("x*y = %d¥n", z);
    printf("x/y = %.1f¥n", w);
    return 0;
}
```

▲ Program3.4　number4.c

```
x*y = 60
x/y = 2.4
```

▲ Result3.4 number4.cの実行結果

Column）intとdoubleの落とし穴

もう1つ、宣言の部分を変えなくても、右のような方法があります。

xの前に(double)と記述されていますね。これは、型変換と呼ばれるもので、一時的に変数xをdouble扱いにする命令になります。

もちろん、「x/(double)y」としてもOKです。yがdouble扱いになるので、計算結果もdoubleになります。

double/int = double

```
#include <stdio.h>

int main(void){
    int x, y, z;
    double w;
    x = 12;    y = 5;    z = x*y;    w = (double)x/y;
    printf("x*y = %d¥n", z);
    printf("x/y = %.1f¥n", w);
    return 0;
}
```

▲ Program3.5　number5.c

```
x*y = 60
x/y = 2.4
```

▲ Result3.5 number5.cの実行結果

#3-3）変数の値

最後にもう1つ。超基本的なことですが、超重要なことです。右のProgram3.6を見てください。実行結果はどうなるでしょうか？

薄々気付いていると思いますが、**1つの変数は1つの値しか持つことができません。**したがって、Program3.6のように、同じ変数に違う値を代入したときは、**最後に代入した値のみが保持されていることになります。**それまでの値は全て破棄されてしまうので、プログラムを書くときはその点に注意してください。

```c
#include <stdio.h>

int main(void){
    int x;
    x = 10;
    x = 20;
    x = 30;
    printf("x = %d¥n", x);
    return 0;
}
```

▲ Program3.6　number6.c

```
x = 30
```

▲ Result3.6 number6.cの実行結果

#3-3）変数の値

　これを踏まえて、右のProgram3.7の実行結果を予想してみましょう。

　「x = x + 20」とか書いてありますね。これはどういう意味でしょうか。

　これは、「元のxの値に20を足した値を、新たにxに代入せよ」という意味です。だから、このような実行結果になるんですね。~~すっごーい!!~~

　失礼しました。今回の講義は以上です。

```c
#include <stdio.h>

int main(void){
    int x;
    x = 10;      ←この時点でxには10が入っている
    x = x + 20;      ←xには30が入っている
    x = x + 30;      ←xには60が入っている
    printf("x = %d¥n", x);
    return 0;
}
```

▲ Program3.7　number7.c

```
x = 60
```

▲ Result3.7 number7.cの実行結果

Question3-1）intとdouble

以下のProgram3.8を実行したときの、実行結果を記せ。

```
#include <stdio.h>

int main(void){
	int price = 50;
	int discount = 15;
	double x = (double)price*(100-discount)/100;
	printf("%d円の%d%%引きは%f円です。¥n", price, discount, x);
}
```

▲ Program3.8　sale.c

Answer3-1）intとdouble

```
#include <stdio.h>

int main(void){
    int price = 50;
    int discount = 15;
    double x = (double)price*(100-discount)/100;
    printf("%d円の%d%%引きは%f円です。¥n", price, discount, x);
}
```

▲ Program3.8　sale.c

```
50円の15%引きは42.500000円です。
```

▲ Result3.8　sale.cの実行結果

xの計算が若干難しかったですね。間違えてしまった人はよく復習しておきましょう。

ちなみに、Program3.6でもやっている通り、変数を宣言した時に値を代入することは可能です。変数宣言時に値を代入することを、変数の初期化と言います。

Question3-2）デバッグ

2つの整数を入力し、それらを除算した計算結果を出力するProgram3.9を作成した。ところが、実際にプログラムを実行させてみると、Result3.9のように誤った計算結果が出力されてしまった。

このような実行結果を回避するように、Program3.9を修正せよ。

```
#include <stdio.h>

int main(void){
        int a, b;    double c;
        scanf("%d", &a);
        scanf("%d", &b);
        c = (double)(a/b);
        printf("a/b = %f¥n", c);
        return 0;
}
```

▲ Program3.9　div.c

```
8 [Enter]
5 [Enter]
a/b = 1.000000
```

▲ Result3.9　div.cの実行結果

Answer3-2）デバッグ

```
#include <stdio.h>

int main(void){
        int a, b;    double c;
        scanf("%d", &a);
        scanf("%d", &b);
        c = (double)(a/b);
        printf("a/b = %f¥n", c);
        return 0;
}
```

▲ Program3.10　div.c

着目すべき点は、左の赤字部分です。

プログラムでも演算順序は一緒なので、このような書き方のとき、最優先で行われる処理は()内の計算となります。つまりa/bが1番最初に行われ、その後に(double)によって型変換されるのです。

しかし、a/bはint/intの計算なので、計算結果は整数になってしまいます。すでに整数になったものを(double)で型変換しても、後の祭りですね……。

したがって、赤字部分を以下のように直せばいいことが分かります。

c = (double)a/b;

Question3-3）除算のプログラム

2つの小数を入力し、それらを除算したときの計算結果を出力するようなプログラムを作成せよ。ただし、計算結果は小数第10位まで表示し、実行結果は下記の【実行例】に準ずるようにせよ。

【実行例】

```
5.9 [Enter]
4.32 [Enter]
1.3657407407
```

double型の変数に対して、値をscanf()から入力するときは、

```
scanf("%lf", &x)
```

というように、変換指定文字を「%lf」にする必要があります。

Answer3-3）除算のプログラム

```
#include <stdio.h>

int main(void){
        double a, b;
        scanf("%lf", &a);
        scanf("%lf", &b);
        printf("%.10f¥n", a/b);
        return 0;
}
```

▲ Program3.11　div2.c

double型を扱う際には、変換指定文字に気を使う必要があります。

printf()するときは、

「%f」「%.1f」「%.2f」etc……

scanf()するときは、

「%lf」

しっかり使い分けてくださいね!!