第2章 「演算と型」のまとめ

2.1 「演算」のまとめ

演算子とオペランド (教 p.24)

とは、「+」,「-」,「*」,「/」のように 演算の働きを持った記号のことである。(教科書 p.221 に C 言語のすべての演算子の表がある。)

＿＿＿＿＿とは、その演算の対象となる式（変数や定数など）のことである。

乗除演算子と加減演算子 (教 p.25)

```
整数 / 整数
```

という演算では、整数としての割算 (小数点以下は＿＿＿＿) の結果が得られる。

```
整数 % 整数
```

では、＿＿＿＿（余り）を求める。

Q 2.1.1 次の式の出力は?
① `printf("%d", 3 / 10)` … ＿＿　　② `printf("%d", 11 % 3)` … ＿＿

printf 関数での「%」文字の表示 (教 p.25)

「`%%`」と 2 つ重ねることで「`%`」という文字そのものを出力することができる。`puts` 関数では、「`%`」はそのまま出力される。

Q 2.1.2 `printf` 関数と `puts` 関数の双方で `100 %` と出力するための式を書け。

① `printf(＿＿＿＿＿＿)`　　② `puts(＿＿＿＿＿＿)`

複数の変換指定 (教 p.27)

変換指定（「`%d`」など）が複数あるときは、第 2 実引数, 第 3 実引数, ...が順に対応する。

Q 2.1.3 次の式の出力は?
① `printf("%d/%d", 10, 24)` … ＿＿＿＿
② `printf("(%d, %d, %d)", 12, 34, 5)` … ＿＿＿＿＿＿

代入演算子 (教 p.29)

「=」演算子は（単純）代入演算子と呼ばれる。

式と代入式 (教 p.29)

変数や定数、それらを演算子で結合したものを＿＿という。

式文 (教 p.29)

式のあとに「;」をつけて文にしたものを式文という。

2.2 「型」のまとめ

「(」～「)」 (教 p.30)

式のなかで先に演算する箇所を示すために丸括弧「(」～「)」を用いる。他の括弧（{, }, [,] など）は別の用途があるため、この目的には使用できない。

整数型と浮動小数点型 (教 p.30)

double 型とは、いわゆる実数 (正確には浮動小数点数) を扱うための型である。もちろん、実数と言っても精度には限界がある。（十進で 15 桁くらい。）

double 型の演算 (教 p.33)

実数を読み込むときは「%lf」を用いる。

```
/* キーボードから実数を変数 fxに読み込む */
scanf("___", &fx);
```

変換指定の使い分け (教 p.33)

以下の表くらいは、この授業の期間中、覚えておくこと。

	int	double
printf	%d	＿＿注
scanf	＿＿	＿＿

注: ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

型と定数 (教 p. 32)

5, 10 などは整数定数、2.718 などは小数点を含むものは浮動小数点定数と呼ばれる。基本的に整数定数は int 型、浮動小数点定数は double 型である。

型と演算 (教 p.34)

```
実数型 / 実数型
```

の演算では、切捨ては行わず、通常の割算が行われる。一方 int と double が混じっている場合、

```
整数型 / 実数型
```

や

```
実数型 / 整数型
```

の場合も、整数 (int) 型のオペランドが＿＿＿＿＿＿が行われて実数 (double) 型になり、double 型の演算となる。

キャスト (cast) (教 p.36)

とは、明示的に＿＿＿＿することである。

```
( 型 ) 式
```

という形で、「式」の値を「型」としての値に変換する。例えば、

```
1  int a, b;
2  …
3  … (double)(a + b) / 2 …
```

では、double 型としての割算が行われる。 (演算子の優先順位に注意する (教 p.221)。割算よりもキャストが先に行われる。)

なお、double から int へのキャスト

```
1  double x = -2.8;
2  printf("%d", (int)x);
```

は切捨てになる。

Q 2.2.1 次の式の出力は?（%f は少数第 6 位まで出力する。）

① `printf("%f", (double)1 / 2)` … ＿＿＿＿＿＿

② `printf("%f", (double)(1 / 2))` … ＿＿＿＿＿＿

変換指定 (教 p.38)

以下のような変換指定は必要に応じて調べれば良い。

説明	例	出力
桁数を揃える	`printf("[%3d]", 1)`	`[  1]`
桁数を揃え先頭を 0 で埋める	`printf("[%03d]", 1)`	`[001]`
小数点以下の桁数を指定する	`printf("[%.3f]", 3.1415926)`	`[3.142]`
16 進数で表示する (小文字)	`printf("[%x]", 127)`	`[7f]`
16 進数で表示する (大文字)	`printf("[%X]", 127)`	`[7F]`

printf 関数が浮動小数点数を出力するとき、ある桁（デフォルトでは小数第 6 位）まで表示して、あとは打ち切るが、この丸め方は C 言語の仕様では特に決められていないので処理系に依存する。一般的には「偶数への丸め」

(bankers' rounding) を採用する処理系が多いようである。つまり、
`printf("%.0f, %.0f\n", 2.5, 3.5)` は "2, 4" と出力する。

Q 2.2.2 次の `printf` 関数の呼出しの出力は?

① `printf("%.4f", 1.0 / 3)` … _____
② `printf("%x", 32)` … __

文法のまとめ

式 (expression)

に以下を追加、

分類	一般形	補足説明
単項演算	単項演算子 式	単項演算子は +, −, &, … など
二項演算	式 二項演算子 式	二項演算子は +, −, *, /, =, … など
かっこ	（ 式 ）	演算の順番を指定するため、 丸括弧で囲んだもの (教 p.30)
浮動小数点数 リテラル		`3.14`, `2.0`, `6.02e23`, `6.6626e-34` など (教 p.32)
キャスト	（ 型 ）式	明示的な型変換 (教 p.36)

文 (statement) とは

これまでのところ、

分類	一般形	補足説明
式文	式 _	式は通常、代入式か関数呼出し[†]（教 p.29）

[†] つまり副作用（値以外の何らかの作用）がある式
