第1章　「まずは慣れよう」のまとめ

1.1　用語のまとめ

教 p.2

コンパイル　とは、ソースファイル（人間が読む／書く形式、C 言語の場合拡張子は ___）を実行ファイル（CPU が直接理解できる形式、Windows 上では拡張子は.exe）などに、翻訳することである。

教 p.3

注釈（コメント）　とは、ソースファイル中の人間向けのメッセージで、コンパイラは無視する部分である。C 言語では ___から ___までが注釈である。さらに新しい C 言語の仕様（C99）では//から行末までという形も利用できる。（Visual Studio Express でも使用可能）

教 p.4

printf　は、表示を行うための関数である。関数とは定義済みのプログラム部品である。関数呼出しは処理の依頼であり、その時に渡すデータを _____という。

教 p.4

文　の末尾には、通常セミコロン「;」が必要である。「{」と「}」の間に置かれた文は上から（同一行に複数文があるときは左から）順次実行される。

教 p.6
教 p.354

printf の変換指定のまとめ　printf の第 1 引数のなかで、%から始まる部分は変換指定と言い、第 2 引数以降の値に順に置き換えられる。整数（10 進数）を表示するための変換指定は ___であり、浮動小数点数の表示は%f を使う。

教 p.9

文字列リテラル　とは、一連の文字を二重引用符 "〜" で囲んだものであり、文字の並びを表す。

教 p.9

拡張表記　\n は _____、\a は警報（ベル）、\t は _____を表す。このような、\を使った書き方を拡張表記という。その他の拡張表記については、教科書 p.234 を参照すること。

教 p.10

変数　とは、数値などのデータを収納するための「箱」（メモリの中の場所）である。C 言語では、変数を使うためには事前に宣言が必要である。

```
int vx;       /* int（整数）型の変数 vxの宣言 */
double fx;    /* double（倍精度浮動小数点数）型の変数 fxの宣言 */
int vx, vy;   /* int（整数）型の変数 vxと vyの宣言 */
```

Q 1.1.1 C 言語の変数にはどのような名前をつけることができるか？(→ 教 p.102)

.....

.....

教 p.11

代入　とは、変数の値を書き換えることである。「変数 =式」という形式で、右辺の式の値を左辺の変数に代入する

教 p.12

初期化子　変数の生成（宣言）のときに値を入れることを初期化という。変数は次の形で初期化することができる。(初期化されないときは “不定値” が入る。)

型名 変数名$_1$ = 初期化子$_1$, ... , 変数名$_n$ = 初期化子$_n$;

初期化子 (initializer) は今のところ “式” (expression) と思っておいて良い。(配列のときに、初期化子と式が異なる場合が出てくる。)

教 p.14

scanf 関数　とは、キーボードから数値などデータを読み込むための関数である。

```
scanf("___", &no);  /* キーボードから整数を変数 noに読み込む */
```

書き方に注意

「&」は「アンパサンド」と読む。詳しくは、ポインタのところで説明する。

教 p.16

puts 関数　は、文字列を出力し、最後に改行を行う。printf のように書式変換は行わない。

1.2 文法のまとめ

式 (expression) とは　これまでのところ、

分類	一般形	補足説明
変数		x, i など
整数リテラル		1, 0, 100, 0xff など
文字列リテラル	"〜"	"Hello\n"など
関数呼出し	関数名 (式, ... , 式)	printf("Hello\n") など

宣言 (declaration) とは　これまでのところ、

分類	一般形	補足説明
変数宣言	型 変数名 = 初期化子 , ... , 変数名 = 初期化子;	型は int, double など

第2章　「演算と型」のまとめ

2.1　用語のまとめ

教 p.22

演算子　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.205 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。

教 p.23

/ 演算子

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 次の式の値は?　　① `3 / 10` ... ＿　② `8 % 3` ... ＿

教 p.23

printf で%文字を表示　「%%」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。

教 p.25

複数の変換指定　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 2.1.2 次の式の出力は?

① `printf("%d/%d", 10, 24)` ... ＿＿＿＿

② `printf("(%d, %d, %d)", 12, 34, 5)` ... ＿＿＿＿＿＿

教 p.27

代入演算子　「=」演算子のことは（単純）代入演算子と呼ばれる。

教 p.27

式　変数や定数、それらを演算子で結合したものを＿＿という。

教 p.27

式文　式のあとに「;」をつけて文にしたものを式文という。

教 p.29

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。

教 p.31

scanf 関数　実数を読み込むときは `%lf` を用いる。

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

教 p.31

変換指定のまとめ　以下の表くらいは、暗記しておくこと。

	int	double
printf	%d	____注
scanf	___	____

注: _____

教 p.32

型と演算

実数 / 実数

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

整数 / 実数

や

実数 / 整数

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。

教 p.34　**キャスト (cast)**　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、

```
int na, nb
...
...  (double)(na + nb) / 2  ...
```

教 p.205　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。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

```
double x = -2.8;
...  (int)x ...
```

は切捨てになる。

Q 2.1.3 次の式の値は?

① (double)1 / 2 ... ___　② (double)(1 / 2) ... ___

教 p.36

高度な変換指定　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]`

Q 2.1.4 次の `printf` 関数の呼出しの出力は?

① `printf("%.4f", 1.0 / 3)` ... _____

② `printf("%x", 32)` ... ___

2.2　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、 括弧で囲んだもの (教 p.28)
浮動小数点数リテラル		`3.14`, `2.0`, `6.02e23`, `6.6626e-34` など (教 p.27)
キャスト	(型) 式	明示的な型変換 (教 p.31)

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.27）