第7章　「基本型」のまとめ

7.1　用語のまとめ

教 p.153

文字型と整数型　signed は＿＿＿＿＿＿、unsigned は＿＿＿＿＿＿の整数を宣言する際の型指定子である。

教 p.154

limits.h ヘッダ　各数値型で表現できる値の最小・最大値をマクロとして集めたヘッダファイルである。

Bcc32 の場合、INT_MIN は＿＿＿＿＿＿＿＿、INT_MAX は＿＿＿＿＿＿＿＿、UINT_MAX は＿＿＿＿＿＿＿＿である。

```
#include <stdio.h>
#include <limits.h>

int main(void) {
    printf("INT_MIN = %d¥n", INT_MIN);
    printf("INT_MAX = %d¥n", INT_MAX);
    printf("UINT_MAX = %ud¥n", UINT_MAX);

    return 0;
}
```

Bcc32 の場合、limits.h はC:¥borland¥bcc55¥Include¥limits.h にあるが、実質的な部分はC:¥borland¥bcc55¥Include¥_lim.h に書かれている。

教 p.156

sizeof 演算子

```
sizeof (型名)
```

という形で指定した型のサイズ（単位:バイト）を返す。

教 p.157

typedef 宣言　typedef 宣言は型の別名をつける。

分類	一般形	補足説明
typedef 宣言	typedef 型 新しい型名 ;	例えば typdef unsigned size_t;

教 p.158

整数定数　8 進定数は先頭に _を、16 進定数は先頭に ___をつけて表記する。

10 進	8 進	16 進
48	060	0x30
65	0101	0x41
97	0141	0x61

教 p.170

整数の表示　printf 関数で整数を 8 進数または 16 進数で表示するためには、それぞれ、___, ___（アルファベットを大文字にしたいときは ___）という書式指定を用いる。

教 p.175

math.h ヘッダ　sin, cos, tan, sqrt （square root — 平方根）, exp, log などの _____ _____ヘッダファイルである。円周率（_____）、自然対数の底（____）などの定数もマクロとして定義されている。

Bcc32 の場合、math.h はC:¥borland¥bcc55¥Include¥math.h にある。

教 p.176

演算子の一覧　優先順位や結合性をすべてを覚える必要はないが、必要に応じて表を調べられるように、どのような演算子があるかくらいは覚えておきたい。(Table 7-4)

教 p.180

sizeof 演算子（その 2）

```
sizeof 式    /* _____ */
```

という形で、式（通常は変数）のサイズ（単位: バイト）を返す。特に、式が配列の場合は配列全体のサイズ（単位: バイト）を返す。

ただし、関数の引数として渡された配列では、別の値（ポインタ型のサイズ）を返すので注意する。

7.2　プログラム例

sizeof 演算子の確認

```
 1:  #include <stdio.h>
 2:
 3:  void foo(int x[]) {
 4:      printf("size=%u¥n", sizeof(x));
 5:  }
 6:
 7:  int main(void) {
 8:      int a[] = { 1, 2, 3 };
 9:
10:      printf("size=%u¥n", sizeof(a));
11:      foo(a);
12:      return 0;
13:  }
```

第8章　「いろいろなプログラムを作ってみよう」のまとめ

8.1　用語のまとめ

教 p.194

再帰 (recursion)　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿。一般に x の定義に x 自身を使用すること。

```
factorial(4)
  →  4 * factorial(3)
    →  4 * 3 * factorial(2)
      →  4 * 3 * 2 * factorial(1)
        →  4 * 3 * 2 * 1 * factorial(0)
          →  4 * 3 * 2 * 1 * 1
```

- 繰返し（for, while）で簡単に実現できることを, 再帰で書くのは（C 言語の場合）良いこととはいえない。階乗の例題プログラムは、あくまでも再帰を説明するためのものと考えること。（もちろん、再帰を使わなければ簡単に書けないプログラムも多い。）
- 再帰関数には、特別な文法も特別な実行規則も必要ない。あくまでも C 言語の普通の関数で、普通の実行規則に基づいて計算される。

教 p.199

getchar 関数　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿関数。

教 p.199

EOF　getchar などが、入力の終わり（＿＿＿＿＿＿＿＿）に達した場合に返す値をマクロで EOF と書く。（stdio.h に定義されている。）

教 p.200

文字　C 言語では文字は、単にその文字に与えられたコード（整数値）で表す。

ASCII コード表での文字コードの抜粋:

文字	10 進	16 進
'0'	48	0x30
'A'	65	0x41
'a'	97	0x61

教 p.203

拡張表記　¥n の他に、¥t, ¥a, ¥b などいくつかの特殊文字を表す表記がある。特に、バックスラッシュ（円記号）そのものを表す時には ¥¥と書く。

教 p.204

putchar 関数　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿。

教 p.205

リダイレクト　標準入出力をファイルへの入出力につなぎかえることで、C 言語ではなく OS (Unix, MS-DOS など) の機能になる。

コマンド名 < ファイル名　— ファイルの内容をコマンドの標準入力に渡す
コマンド名 > ファイル名　— コマンドの標準出力をファイルに書込む
コマンド名 >> ファイル名　— コマンドの標準出力をファイルの最後に追加する形で書込む

8.2 プログラム例

ハノイの塔（再帰）

```
 1:  #include <stdio.h>
 2:
 3:  void move(int n, int a, int b) {
 4:      printf("ディスク%d を棒%d から棒%d へ¥n", n, a, b);
 5:  }
 6:
 7:  /* n 枚のディスクを a から b に移動する手順 */
 8:  void hanoi(int n, int a, int b, int c) {
 9:      if (n>0) {
10:          hanoi(n-1, a, c, b);
11:          move(n, a, b);
12:          hanoi(n-1, c, b, a);
13:      }
14:  }
15:
16:  int main(void) {
17:      int n;
18:      printf("円盤は何枚ですか? "); scanf("%d", &n);
19:      hanoi(n, 1, 2, 3);
20:      return 0;
21:  }
```

樹の描画（再帰）

```
 1:  #include <stdio.h>
 2:  #include <math.h>
 3:
 4:  void drawTree(int d, double x, double y, double r, double t) {
 5:      /* d  --- 再帰の深さ、(x, y)  --- 枝の根元の座標              */
 6:      /* r  --- 枝の長さ、  t       --- 枝の伸びる向き（ラジアン） */
 7:      double r1;
 8:      if (d == 0) return; /* 打切り */
 9:
10:      printf("%6.3f %6.3f %6.3f %6.3f¥n",
11:              x,    y,    x + r * cos(t), y + r * sin(t));
12:      drawTree(d-1, x + r  * cos(t), y + r  * sin(t), 0.5 * r, t);
13:      r1 = 0.5 * r;
14:      drawTree(d-1, x + r1 * cos(t), y + r1 * sin(t), 0.5 * r, t + M_PI/2);
15:      drawTree(d-1, x + r1 * cos(t), y + r1 * sin(t), 0.5 * r, t - M_PI/2);
16:  }
17:
18:  int main(void) {
19:      drawTree(6, 128, 255, 128,  -M_PI / 2);
20:      return 0;
21:  }
```