

第7章 「基本型」のまとめ

7.2 「整数型と文字型」のまとめ

整数型と文字型 (教 p.186)

signed は _____ (負の数も扱える) 整数、unsigned は _____ (正の数と 0 のみ扱える) 整数を宣言する際の型指定子である。short と long はそれぞれ、標準よりも狭い、広い範囲を扱える (かもしれない) 整数を宣言する際の型指定子である。printf 関数での signed, long に対する変換指定はそれぞれ _____, _____ である。

<limits.h> ヘッダー (教 p.188)

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ーである。

Q 7.2.1 教科書 List 7-1 を実行し、次の空欄を埋めよ。

『 _____ の場合、INT_MIN は _____、INT_MAX は _____、UINT_MAX は _____ である。』

sizeof 演算子 (教 p.192)

```
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。

```
sizeof 式 /* 通常は上の形にあわせて式を ( ~ ) で囲む */
```

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。

Q 7.2.2 配列 a の要素数を表す sizeof 演算子を使った式は?

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

Q 7.2.3 次の sizeof 演算子を使用するプログラム (sizeof.c) の出力はどうなるか? 実行して確かめよ。

```
1 #include <stdio.h>
2
3 void foo(int x[]) {
4     printf("sizeof(x) = %u\n", (unsigned) sizeof(x));
5 }
```

```

6
7 int main(void) {
8     int a[] = { 1, 2, 3 };
9     int b[] = { 4, 5, 6, 7 };
10
11     printf("sizeof(a) = %u\n", (unsigned)sizeof(a));
12     printf("sizeof(b) = %u\n", (unsigned)sizeof(b));
13     foo(a);
14     foo(b);
15     return 0;
16 }

```

(____ の場合)

sizeof(a) = ____
 sizeof(b) = ____
 sizeof(x) = ____
 sizeof(x) = ____

関数の引数として渡される配列に対する sizeof はポインター型（第 10 章）のサイズを返す。

size_t 型と typedef 宣言 (教 p.194)

typedef 宣言は型の別名をつける。構造体・ポインターを学習したあとは頻繁に使う。

分類	一般形
typedef 宣言	typedef 型 新しい型名 ;

例えば typedef unsigned size_t; と宣言すると、size_t が unsigned の別名となる。

ビット単位の論理演算 (教 p.202)

ビット単位の論理演算・シフト演算などは組み込み用途では多用される。必要に応じて調べられるようにしておく。

整数定数 (教 p.210)

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

整数の表示 (教 p.212)

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、____, ____ (A ~ F を大文字にしたいときは ____) という書式指定を用いる。

7.3 「浮動小数点型」のまとめ

math.h ヘッダー (教 p.217)

`sin, cos, tan, sqrt` (square root — 平方根), `exp, log` などの [数学関数のプロトタイプ宣言が集められている](#) ヘッダーである。

gcc の場合、`math.h` ヘッダーの関数を使ったプログラムをコンパイルするとき、大抵 `-lm` オプションが必要である。

繰返しの制御 (教 p.218)

誤差が累積するので、繰返しを制御する変数に、できるだけ浮動小数点数型 (`float, double`) は使わない。

Q 7.3.1 次のプログラム (の断片) の出力はどうなるか? (なお `cl, gcc` と同じ結果になることを確認しています。)

```
1 printf("%.60f", 0.1);
```

7.4 「演算と演算子」のまとめ

演算子の優先順位と結合性 (教 p.221)

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 7-13)