

第9章 「文字列の基本」のまとめ

9.1 用語のまとめ

文字列リテラル 文字の並びを _____ (") で囲んだもの。

内部的には char 型の配列として表現される。ただし、末尾に終わりを表すために _____ ('\0') が自動的に付加される。

文字列リテラル中に二重引用符 (") を入れる場合は、 ___ という拡張表記を用いる。

ナル (ヌル) 文字 ('\0') 0 という値を持つ文字のこと。

'\0' (ナル文字 — ASCII コード _) と '0' (数字の 0 — ASCII コード _) は全く異なるものである。

文字列 C 言語では文字列は文字 (char) 型の配列で、終わりをナル文字で表したものである。

文字列を printf で出力するための変換指定は ___ である。

文字列 (配列) の初期化 文字列を格納する配列は次のように初期化できる。

```
char str[4] = { 'A', 'B', 'C', '\0' }; /* 要素数の 4 は省略可能 */
/* または */
char str[4] = "ABC";                /* 要素数の 4 は省略可能
                                     (ただし 3 ではないことに注意する) */
```

文字列の読み込み scanf の変換指定には %s を使い、読み込む配列には & を付けずに渡す。

```
chr str[40];
printf("文字列を入力して下さい: "); scanf("%s", _____);
```

実用的なプログラムでは文字列を入力するために、scanf を使用するのには推奨できない。予測よりも長い文字列を入力されると、プログラムが暴走する可能性がある。これは _____ というセキュリティホールになる。この問題を回避するために fgets という関数を使うことが多い

教 p.208

教 p.208

教 p.210

教 p.211

教 p.212

fgetstest.c

```
#include <stdio.h>

int main(void) {
    char str[40];

    printf("文字列を入力して下さい: "); fgets(str, 40, stdin);
    printf("%s", str);
    return 0;
}
```

例えば上の例で `scanf("%s", str);` の代わりに `fgets(str, 40, stdin);` とすると、最大 39 (=40-1) 文字まで (もしくは改行文字まで) 読み込んで最後にナル文字を追加する。

一般に、`fgets` の第 1 引数は文字列を読み込む先の `char` の配列で、第 2 引数は最大読み込む文字数+1 (ナル文字の分)、第 3 引数は読み込む元である。上の例では標準入力 (`stdin`) を指定しているが、ファイルなどからも読み込むことができる。

この方式は安全だが、大抵の場合、最後に読み込まれる改行文字を取り除く必要があるので、ちょっとだけ面倒である。(改行文字を取り除くには、通常、`sscanf` という関数を使う。) このため以下では、あくまでも練習用と割り切って `scanf("%s", ...)` というかたちで、文字列を読み込むことがある。

教 p.214

文字列の配列 もちろん、文字列の配列をつくることも可能である。

```
char cs[][10] = {"Hayashi", "Takamatsu", "Kagawa"}
```

これは、次のように宣言するのと同じ事である。

```
char cs[][10] = { {'H', 'a', 'y', 'a', 's', 'h', 'i', 0, 0, 0},
                  {'T', 'a', 'k', 'a', 'm', 'a', 't', 's', 'u', 0},
                  {'K', 'a', 'g', 'a', 'w', 'a', 0, 0, 0, 0} };
```

(空いている後ろの部分はナル文字で初期化される。)

教 pp.216~219

文字列の操作 文字列を扱うプログラムは、ナル文字が出現するまでループする、という形になることが多い。

```
int my_strlen(const char str[]) {
    int len = _;

    _____
    _____
    -
    return len;
}
```

教 p.222

大文字・小文字の変換 大文字・小文字の変換には、_____ というヘッダファイルで宣言されている `int toupper(int c)` 関数、`int tolower(int c)` 関数を利用することができる。

(`ctype.h` には文字のテスト・変換に使用できる関数がいくつか用意されている。)