第9章　「文字列の基本」のまとめ

9.1　用語のまとめ

教 p.208

文字列リテラル　文字の並びを二重引用符（"）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ナル（ヌル）文字（¥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、＿＿(空欄 9.1.1)という拡張表記を用いる。

Q 9.1.1 "Hello" と（二重引用符を含めて）出力する printf の呼出し式を書け。
答: ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

教 p.208

ナル（ヌル）文字（'¥0'）　0 という値（ASCII コード）を持つ文字のこと。

'¥0'（ナル文字 — ASCII コード ＿＿(空欄 9.1.2)）と'0'（数字の 0 — ASCII コード ＿＿(空欄 9.1.3)）は全く異なるものである。

教 p.210

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%s である。

教 p.211

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。

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

教 p.212

空文字列　内容が空の文字列は次のように宣言することができる。

```
char ns[] = "";
```

これは、次のような宣言と同等である。

```
char ns[] = { '\0' }
```

（空の配列ではないので注意すること。）

Q 9.1.2 次の宣言の中で同じ内容に初期化される配列をグループ化せよ。

```
1   char a[] = "";           答: ___
2   char b[] = "0";          答: ___
3   char c[] = "\0";         答: ___
4   char d[] = { 0 };        答: ___
5   char e[] = { '0' };      答: ___
6   char f[] = { '\0' };     答: ___
7   char g[] = { 0, '\0' };  答: ___
8   char h[] = { '0', 0 };   答: ___
9   char i[] = { '\0', 0 };  答: ___
10  char i[] = { 48, 0 };    答: ___
```

教 p.212

文字列の読み込み　scanf の変換指定には %s を用い、読み込む配列には&を**付けずに**渡す。

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

注意: 実用する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。これはバッファーオーバーフローという**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

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。

ファイル名: fgetstest.c

```
1   #include <stdio.h>
2
3   int main(void) {
4       char str[40];
5
6       printf("文字列を入力して下さい:␣");
7       fgets(str, 40, stdin);
8       printf("%s", str);
9       return 0;
10  }
```

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。

Q 9.1.3 入力された文字列をオウム返しに出力するプログラムを次のように作成する。空欄を埋めてプログラムを完成させよ。

```
1  #include <stdio.h>
2
3  int main(void) {
4      char str[40];
5
6      printf("文字列を入力して下さい:␣");
7      scanf("__", ____);
8      printf("あなたは「__」と入力しました。\n", ____);
9      return 0;
10 }
```

教 p.214

文字列の配列　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ことである。

```
1  char cs[][10] =
2      { { 'H', 'a', 'y', 'a', 's', 'h', 'i',  0,   0,   0 },
3        { 'T', 'a', 'k', 'a', 'm', 'a', 't', 's', 'u',  0 },
4        { '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。

```
1  int my_strlen(const char str[]) {
2      int len = _（空欄 9.1.4）;
3
4                                          （空欄 9.1.5）
5      ____________________________________
6                （空欄 9.1.6）
7        （空欄 9.1.7）
8      _
9      return len;
10 }
```

教 p.222

大文字・小文字の変換　大文字・小文字の変換には、ctype.h 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。）