プログラミング演習
～文字列～

1　目的

C 言語における文字列の基本的な使い方を理解する．

2　製作対象 文字列とは

変数の中で，英数字特殊文字 1 文字を記憶する変数は，char 型である．特殊文字とは，カッコ”(”やスペース・改行などである．例えば以下では char 型の変数 c は 1 文字 a を記憶している．

- char c;
- c=’a’;

一般の文章，例えば ”My name is hoge.”など，は 1 文字が集まって列となしたものである．そこで 1 文字 (char 型) を複数記憶できる配列 (char 型の配列) を考えると単語や文章などを記憶できると考える．このようにして考えている文字型配列を文字列，という．ここで，変数への大入などで 1 文字を表す場合には，’(シングルクォーテーション) で値がくくられ，文字列を表す場合には，”(ダブルクォーテーション) でくくられる．

ここで，int 型の配列のように他の配列と文字配列 (文字列) の大きな違いである，”列の終わり”について述べる．int 型の配列に，3,4,10,15... と記憶されている場合，よく問題になるのは”何個の数値が記憶されているか”である．一方文字列では，”何個の文字が記憶されているか”ではなく，”どこが文字列 (文章) の終わりなのか”が重要になる．文字を取り扱う関数の数も多く，この文字列特有の特徴から，”文字列 (文章) の終わり”を表す特別な記号が定義されている．それが’\0’(これで 1 文字) である．ちなみに，普通の文章では，”。”や”．”が文字の終わりを表す特別な記号であるが，この記号も”文字”の 1 つなので，コンピュータ独自の’\0’ が使われている．

例として，”word”という文字列 (文章はスペースを含み，スペースは表現しにくいので単語にする) を考えると，これは’w’ と’o’ と’r’ と’d’ と’\0’ の文字変数の集まりである．また，’a’ と”a”の違いは，前者は’a’ という文字であるのに対して，”a”は’a’ と’\0’ の文字変数の集まりである．

2.2 プログラムは文字列の基本的な使い方を示したものである．特に，15 行目と 17 行目の宣言の仕方が独特であり，15 行目は宣言と同時に各配列の要素に文字を代入している．17 行目は，15 行目の処理に加え，配列の要素数を自動的に決定している．この場合，文字の数 20 文字に加え終了文字’\0’ があるので計 21 文字となるので，

- char string3[21]=...;

と同じとなる．

また使い方として 39 行目から 41 行目のように”文字列”として使うこともでき，配列の名前を使って表示している (配列の復習:名前はポインタ変数であるので，printf 関数の中ではポインタ変数として処理を行っている)．

2.1 準備

ディレクトリ ”programming12”を作成する．今回の演習では，プログラムの作成や必要ファイルのダウンロードは，programming12 ディレクトリで行う．

2.2 プログラム

```
1  /*********************************
2        文字列の基本的な使い方
3  *********************************/
4
5  #include <stdio.h>
6
7  int main(){
8    int loop;
9
10   /* 宣言 &  値の代入の仕方 */
11
12   /* 宣言のみ */
13   char string1[100];
14   /* 宣言+代入 */
15   char string2[100]="My name is Iron Boy.";
16   /* 宣言+代入 (配列の要素数は自動的に決まる) */
17   char string3[]="My number is 2100000";
18
19   /* 代入 */
20   string1[0] = 'S';
21   string1[1] = 't';
22   string1[2] = 'a';
23   string1[3] = 'r';
24   string1[4] = 't';
25   string1[5] = '\0';
26
27   /* 表示 1 文字毎 */
28   for(loop=0 ; string1[loop]!='\0' ; loop++){
29     printf("string1[%d]=%c\n",loop,string1[loop]);
30   }
31   for(loop=0 ; string2[loop]!='\0' ; loop++){
```

```
32        printf("string2[%d]=%c\n",loop,string2[loop]);
33      }
34      for(loop=0 ; string3[loop]!='\0' ; loop++){
35        printf("string3[%d]=%c\n",loop,string3[loop]);
36      }
37
38      /* 表示　文字列として */
39      printf("string1=%s\n",string1);
40      printf("string2=%s\n",string2);
41      printf("string3=%s\n",string3);
42
43      return 0;
44    }
```

2.3　コンパイル

上記のプログラムを作成し，ファイル名 string-sample.c として保存してコンパイルを行う．コンパイル後の名前を string-sample とする．

> gcc -o string-sample string-sample.c

2.4　動作実験

作成したプログラムを実行させ，動作を理解しよう．

2.5　調べてみよう

以下のように文字列を宣言することもできるが，動作が他の宣言と異なる．どのように異なり，何故異なるか調べて考えてみよう．

- char* string4 = ”This is the fixed string.”;

2.6　やってみよう

2.2 プログラムの 42 行目に自作プログラムを追加し，名前，学籍番号を”scanf で入力して表示するプログラム”にしてみよう