[プログラミング入門Ｉ　（５）］ 文字と文字列

文字型と文字コード

文字列とは、文字の列です。文字のデータ型が文字型です。文字型は、char で宣言します。例えば、文字型の変数 c は、以下のように宣言します。

```
char c;
```

文字型の定数は、'１文字'と書きます。例えば、Ａという文字は、'A'です。変数 c に A をいれるには、

```
c = 'A';
```

とかきます。文字を 1 文字だけ出力するには、%c をつかいます。

```
printf("chacter is %c",c);
```

scanf でも同じです。

```
scanf("%c",&c);
```

これをやるために、別の関数 getchar と putchar を使うこともできます。こちらは、関数ですので、入力する場合は、

```
C = getchar();
```

出力する場合には、

```
putchar(c);
```

というようにして使います。

実際、計算機の中では文字は数字で表現されています。これを**文字コード**といいます。この対応はいわば約束ごとで、現在ほとんどの計算機で使われているのが ASCII コードと言われるものです（端末で man ascii とコマンドを打てば、コード表が見られます)。実は数字なので、この表にしたがって、足し算や比較ができます。例えば、'A'から'Z'までは順番に並んでいるので、'A'+1 は、コード表から'B'になります。大文字のアルファベットであるかどうかを判定するには、

```
c >= 'A' && c <= 'Z'
```

で判定することができます。また、小文字の'a'から'z'までも順番に並んでいるので、たとえば、'a'から‘ｚ’の小文字 c を、大文字に変換するには、以下の文でできます。

```
c = c-'a'+'A';
```

また、数字の文字も'0'から'9'まで順番に並んでいるので、数字の文字を数字に直すのは'0'を引くことによって、求めることができます。たとえば、'5'-'0'は、5 になります。

文字列とは

文字列は、文字の列ですので、実は文字の１次元配列です。

```
char s[10];
```

これは、10 文字からなる文字列です。文字列の入出力は、%s という書式を使います。

```
scanf("%s",s);      printf("string is %s¥n",s);
```

scanf では、&がつかないことに注意してください。scanf は空白までの文字列を入力します。

文字列は文字の配列であると述べましたが、正確にはＣ言語では文字列とは「コード 0 で終わる文字列」です。コード 0 は文字列の最後を示すコードとして使われます。なので、１０文字の文字列 s に文字 ABC を入力すると、配列の最初の 4 要素に'A','B','C','¥0',が入ります（'¥0'は、コード 0 をあらわす文字定数)。printf の%s では、0 までの文字を出力します。文字列のための文字配列は文字列の長さ+1 の要素が必要であることに注意してください。

printf に使っている"…"で囲まれる文字列は文字列定数です。文字列の宣言のところで、

```
char s[10] = "ABC";
```

と書いて、初期値を代入することができます。これは宣言のところだけできますが（初期化)、代入文ではできません。

では練習問題として、文字中の小文字を大文字に変換してプリントするプログラムを考えてみましょう。

```
main(){
    int i;
    char s[10];
    printf("please input string?");
    scanf("%s",s);
    for(i = 0; s[i] != '¥0'; i++)
      if(s[i] >= 'a' && s[i] <= 'z') s[i] = s[i] – 'a'+'Z';
```

```
        printf("result = %s¥n",s);
        return 0;
    }
```
ます、メッセージをプリントアウトした後、scanf で文字列を s に読み込みます。i = 0 から、順番に、s[i]が'¥0'になるまで、繰り返します。繰り返しでは、s[i]が小文字である場合には、文字コードのところで説明した方法で変換します。
次に、文字列 char s1[10]から char s2[10]にコピーするループを示します。
for(i = 0; s1[i] != '¥0'; i++) s2[i] = s1[i];
s2[i] = '¥0';
コピーされるほうの文字列を順番にアクセスしていき、s1[i]が'¥0'、つまり、文字列の最後になるまで、順にコピーをしています。最後に s2 の最後に'¥0'をいれなくてはいけないことに注意してください。
とにかく、文字列は最後の文字が'¥0'の文字の配列であることをおぼえておきましょう。

配列の引数

さて、前回関数を説明しました。式で計算した値だけでなく、配列を関数に引数として渡すためには、次のように書きます。

関数の返す値のデータ型　関数名（配列パラメータのデータ型　配列のパラメータ名[]、....）
{　...後は同じ...　}

関数宣言でパラメータ宣言のところにでは、サイズなしで[]をつけておきます。関数の中では、パラメータで宣言された配列名は配列と同じように A[i]というように配列と同じように使うことができます。
呼び出し側では、次のように配列名を引数にします。

関数名（引数の配列名、....）

文字列も文字の配列ですから、同じように関数に引き渡すことができます。配列については後で説明するポインタと深く関連しています。
たとえば、配列とその配列サイズを引数として、配列の要素の加算した結果を返す関数 sum は以下のようになります

```
    int sum(int a[],int n){
      int i,s;
      s = 0;
      for(i = 0; i < n; i++) s+= a[i];
      return s;
    }
```
これを使って、ためしに文字列をコピーする関数を作ってみることにしましょう。
```
    void string_copy(char s2[],char s1[]){
        int i;
        for(i = 0; s1[i] != '¥0'; i++) s2[i] = s1[i];
        s2[i] = '¥0';
    }
```
やっている内容はすでに説明したものです。

文字列を扱うライブラリ関数

便利な関数として、文字列のコピー関数 strcpy、連結 strcat、辞書順比較 strcmp があります。
strcpy(文字列 1,文字列２)は、文字列１に文字列２をコピーします。やっている内容は上に示したものです。strcmp(文字列１、文字列２)は、文字列１と文字列２を辞書順に比較して、前であれば、-1、同じであれば、0、後ろであれば、1 を返します。たとえば、同じ文字列かどうかを判定するには、0 と比較します。
```
  strcmp(s1,s2) == 0
```
また、strcat は、strcat(文字列 1,文字列２)と呼び出した時に、文字列１の最後に文字列２の内容をコピーします。

なお、日本語の文字列については、各自教科書などで勉強してください。