# 文字列の取り扱い

### *演習問題 10.1　文法事項*

1）C 言語の文字列の場合、長さや文字列の終わりを示すのにどのような方法を用いていますか。簡潔に答えなさい。

**解答例**　C 言語には文字列を直接取り扱うデータ型は用意されておらず、文字列定数の他は、**char** 型の配列に文字の並びを格納する方法で文字列を取り扱います。いずれの場合も、文字の並びの最後に **'\0'**( ヌル文字 ) を付加して文字列の終端を表します。

2）文字型の 1 次元配列に、文字列を格納する場合の注意する事項を答えなさい。

**解答例**　文字列には終端を表す **'\0'**( ヌル文字 ) が付加されるため、文字列を格納する配列の要素数は、文字列の長さより 1 だけ多い必要があります。

3）C 言語には、文字列に対応した代入演算子がないため、文字列の複写を直接行うことはできません。文字列を複写する方法について簡潔に答えなさい。

**解答例**　文字列の複写は、文字列が格納された配列要素の内容を代入演算子を用いて 1 つずつ終端記号まで複写する方法で行うことができます。この操作は、配列を直接用いて記述する代わりに、ポインタを用いて記述するとスマートに記述することができます。( 注：ポインタに関しては第 11 章で学びます。)

このような文字列の複写は、自分でプログラミングしてもたいした労力ではありません。しかし、標準関数として文字列の複写を行う **strcpy()** 関数や **strncpy()** 関数などが用意されているので、それらを用いる方が手間がかからず安全で良いでしょう。

### *演習問題 10.2*

1）1 つの文字列と、1 つの文字を入力し、文字列中にその文字が何回含まれているかを調べるプログラムを作りなさい。

**解答例**　解答例を解答プログラム 10.1 に示します。

**for** 文によって、配列 **string[]** に格納された文字列中に、変数 **letter** に格納された文字が幾つあるかを、変数 **count** を使って数えています。このために **string[i]** と **letter** が一致した時だけ、変数 **count** をインクリメントしています。この **for** 文は、**string[i]** の内容が **'\0'**( ヌル文字 ) でない限り繰り返されるので、文字列の最後までチェックされることになります。

このような繰り返しでは、宣言された配列の範囲を超えてアクセスすることを避けなければなりません。そこで、繰り返しの条件として、配列の要素番号が宣言した要素数未満であることをチェックし、これが成り立たないときには繰り返しを終了するようにします。こうしておけば、文字列に不備があった場合にも停止する安全なプログラムになります。

**for** 文の第 1 フィールドに、「**,**( カンマ )」で区切って、複数の式を書ける点にも注目して下さい。

**解答プログラム 10.1**　**文字列中の文字の検索**

```
/************************************************
    p10_1.c
    文字列中の文字の探索
    Tadaaki Shimizu    2006.11.21
************************************************/
#include <stdio.h>
#define STR_LENGTH 256 // 文字列の最大長

main()
{
    char string[STR_LENGTH];    // 文字列
    char letter;                // 探索する文字
    int  count;                 // 文字が含まれる回数
    int  i;                     // カウンタ変数

    // 文字列と文字の入力
    printf(" 文字列の入力 : ");
    scanf("%s", string);
    printf(" 探索する文字の入力 : ");
    scanf("\n");    // 1 文字入力のためのおまじない
    scanf("%c", &letter);

    for(i = 0, count = 0; i < STR_LENGTH && string[i] != '\0'; i++)
        if(string[i] == letter) count++;

    printf(" [%c] は %d 回含まれています。\n", letter, count);
    exit(0);
}
```

2） 1 ) のプログラムを、改良し、文字列中の何文字目に、指定された文字が含まれているのかを答えるプログラムを作りなさい。

**解答例** 解答例を解答プログラム 10.2 に示します。

このプログラムは、解答プログラム 10.1 とほとんど同じです。文字が含まれる位置を記憶するために、**int** 型の配列 **position[]** を用いています。文字の位置の表示を文字の探索の **for** 文の中で行ってしまえば、配列 **position[]** は必要なく、プログラムも単純になりますが、文字の探索と結果の表示を切り分けるために、わざとこのようなプログラムにしてみました。

C 言語の配列は、**0** から番号付けされますが、一般に「何番目」と聞かれれば 1 から数えるので、位置の情報に **+1** をしています。

**解答プログラム 10.2　文字列中の文字の探索**

```
/************************************************
    p10_2.c
    文字列中の文字の探索２
    Tadaaki Shimizu    2006.11.21
************************************************/
#include <stdio.h>
#define STR_LENGTH 256 // 文字列の最大長

main()
{
    char string[STR_LENGTH];    // 文字列
    char letter;                // 探索する文字
    int  count;                 // 文字が含まれる回数
    int  position[STR_LENGTH]; // 文字が含まれる位置
    int  i;                     // カウンタ変数

    // 文字列と文字の入力
    printf(" 文字列の入力 : ");
    scanf("%s", string);
    printf(" 探索する文字の入力 : ");
    scanf("\n");    // 1 文字入力のためのおまじない
    scanf("%c", &letter);

    for(i = 0, count = 0; i < STR_LENGTH && string[i] != '\0'; i++) {
        if(string[i] == letter) {
            position[count] = i + 1;
            count++;
        }
    }

    printf(" [%c] は %d 回含まれています。\n", letter, count);
    printf(" 含まれる位置は、\n");
    for(i = 0; i < count; i++)
        printf(" %d  ", position[i]);
    printf("\n の位置です。\n");
    exit(0);
}
```

### *演習問題 10.3*

入力された文字列中の文字を、逆順に複写し、出力するプログラムを書きなさい。

**解答例**　本文中のプログラム例 10.3 がよく理解できていれば、簡単な問題です。解答例を解答プログラム 10.3 に示します。

文字列を逆順に複写するために、配列 **aStr[]** に入力された文字列の末尾をみつけることで、その長さを調べます。これを用いて、**aStr[]** に格納された文字列の末尾の文字から順に、**bStr[]** の先頭から複写していきます。

### *演習問題 10.4*

プログラム例10.6を改良し、1から10までの数に対する英単語を答えるプログラムにしなさい。また、この範囲外の入力があった場合に、何らかのエラー処理を行うようにしなさい。

**解答例**　解答例を解答プログラム 10.4 に示します。

このプログラムは、プログラム例 10.6 の 2 次元配列の初期化宣言で数詞を増やし、エラー処理を加えただけのものです。数詞を増やすためには、2 次元の **char** 型配

**解答プログラム 10.3**　文字列の逆順コピー

```
/************************************************
    p10_3.c
    文字列の逆順コピー
    Tadaaki Shimizu    2006.11.21
************************************************/
#include <stdio.h>
#define STR_LENGTH 256 // 文字列の最大長さ

main()
{
    char aStr[STR_LENGTH];  // 入力される文字列
    char bStr[STR_LENGTH];  // 複写先の文字列
    int  length;            // 文字列の長さ
    int  i;                 // カウンタ変数

    // 文字列の入力
    printf(" Input string >> ");
    scanf("%s", aStr);

    // 末尾の発見
    for(length = 0; length < STR_LENGTH && aStr[length] != '\0'; length++)
        ; // 空文
    // 逆順コピー
    for(i = 0; i < length; i++)
        bStr[i] = aStr[length - i -1];
    bStr[i] = '\0';

    printf("Input :%s\n", aStr);
    printf("copy  :%s\n", bStr);
    exit(0);
}
```

列の初期化付き宣言で、数詞の文字列を増やすだけでよく、処理の方は全く変わらないことに注目してください。

このような手法は、エラーメッセージのメッセージを表示する必要があるプログラムでよく用いられます。例えば、幾つものエラーメッセージを 2 次元の **char** 型配列にまとめておき、整数値で表されたエラーコードによって、解答プログラム 10.4 と同じ方法で、表示するメッセーージを切り替えるのです。表示すべきメッセージが増えたり、メッセージの内容に変更があった場合にも、配列の宣言部分を変更するだけで簡単に対処できます。

また、このようにプログラム中で変更されず、表示などの処理だけに使われる配列は、下記のように宣言しておくのがよいでしょう。

```
const char numName[10][6] = { "one", "two", ... // 以下略
```

このように、**const** を付加して宣言しておけば、プログラムのミスなどで不正な内容変更が行われるような場合には、コンパイラがワーニングを出して教えてくれます。

**解答プログラム 10.4　数値を数詞に変換するプログラム**

```
/************************************************
    p10_4.c
    入力された整数に対応する英単語を答えるプログラム
              2 次元の文字型配列の応用
    Tadaaki Shimizu    2006.11.21
************************************************/
#include <stdio.h>

main()
{
    char numName[10][6] = {"one", "two", "three", "four",
                           "five", "six", "seven", "eight",
                           "nine", "ten"};     // 英数詞
    int  number;                               // 入力される数

   // 数値入力
    printf(" 整数値入力 (1-10) >> ");
    scanf("%d", &number);
    if(number < 1 || number > 10) {
        printf(" ERROR: 入力された数値が範囲を超えています。\n");
        exit(0);
    }

    printf("%s\n", numName[number-1]);
    exit(0);
}
```