文字列

コンピュータ内部での文字の表現

英数字の場合

コンピュータ内部では全データは数値として扱われる。そのため、文字には対応する数値が割り振られており、これを基にしてプログラムは動作する。文字と数値の対応表にはアスキーコード表というものがあり、この表では英数字と記号に 0～127 の整数がそれぞれ割り当てられている。以下に一例を示す。

表１．アスキーコードの一例

文字	16 進数	10 進数
a	0x61	97
A	0x41	65
+	0x2B	43
1	0x31	49

プログラムでアスキーコードの値を確かめたければ、以下のようにすればよい。

プログラム例１

```
#include <stdio.h>

int main(void)
{
        char c;

        c = 'L';
        printf("L = %d¥n",  c);

        return  0;
}
```

出力

```
L = 76
```

文字型変数(char)は１バイト(8 ビット)のデータが格納でき、格納できる値は-127～128 までの整数である。そのため、文字型変数１つで表現できるのは最大 256 種類の文字となる。

日本語の場合

日本語のように多くの文字を使う際には 256 種類しか表現できないコードでは対応出来ない。そのため、日本語はアスキーコードとは異なる文字コードが使われる。代表的なのが、

日本語 euc	extend unix code の略。主に linux、UNIX で使われている。
shift-jis	主に Windows、machintosh で使われている。
JIS コード	日本工業規格 JIS(Japanse Industrial Standards)が定めたコード。
Unicode	全ての文字の対応付けを行っている。

である。例えば、「秋」という漢字は、表２のようになっている。

表２．各コードでの"秋"。C 言語の場合、先頭に 0x とつく数字は 16 進数を表す。

コード	16 進数	10 進数
日本語 EUC	0xbda9	48553
shift-jis	0x8f48	36680
jis コード	0xbda9	48553
Unicode	0x79cb	31179

英数字と日本語では 1 文字を表現するために必要な情報量が異なる。英数字は 1 バイトで十分であるが、日本語は 2 バイト必要である（Unicode を除く）。

文字の記憶方法

英数字 1 文字の場合（文字型変数）

英数字 1 文字の場合、文字型変数を用いる。次のように宣言をすれば、変数名 a の文字を入れる箱が用意される。この箱の容量は 1 バイトである。

```
char a;
```

型名の char と言うのは、character(文字)の略である。この変数に文字を格納するのは簡単で、

```
a = 'A';
```

とすればよい。格納したい文字をシングルクォーテーションで囲み、代入演算子(=)を使う。文字型変数のイメージを図 1 に示す。

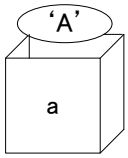

図 1　文字型変数 a を用意して、' A'を格納。この箱の大きさは 1 バイトなので、日本語を入れることはできない。

表示まで含めた文字型変数を使ったプログラムは、次のようになる。

プログラム例２

```
#include <stdio.h>

int main( void )
{
        char a;                 // 文字型の変数
        a =  'A' ;              // 代入
        printf("%c¥n", a);      // 表示

        return 0;
}
```

出力

```
A
```

英数字の文字列の場合（文字型配列）

文字型配列の使い方

次に、いくつかの英数字で構成される文字列の場合について述べる。このようなときは文字型の配列のデータ構造を使う。以前、同じ型の数値データが複数ある場合、int 型あるいは double 型の配列を使ったのと同じである。文字型の配列を使うときには次のように宣言する。

```
char  a[10];
```

これで 10 個の文字を格納できるメモリの領域が確保できる。しかし、実際に入れることができる文字数は 9 文字である。文字列の最後の文字の後に、文字列終了の記号を入れることになるため、1 文字分少なくなる。文字列終了の記号は'¥0'という特殊文字が使われ、これを NULL 文字と言う。

いろいろな方法で、文字型配列に文字を格納することができる。まずは、配列の要素毎に文字を

入れる方法を示す。

プログラム例３

```
#include <stdio.h>

int main( void )
{
        char a[6];              // 文字型の配列

        a[0] = 'A';
        a[1] = 'k';
        a[2] = 'i';
        a[3] = 't';
        a[4] = 'a';
        a[5] = '¥0';

        printf( "%s¥n", a );    // 表示

        return 0;
}
```

出力

```
Akita
```

実際、この方法で文字を代入することはまれであるが、文字列のある特定の要素を操作するには有効である。変数の初期化であれば、以下のような書き方となる。

```
char a[10] = {'A', 'k', 'i', 't', 'a', '¥0'};
```

これは配列の初期化の方法で習った初期化方法である。文字列については以下のような記述も許されている。

```
char a[10] = "Akita";
```

文字列の終了を表す NULL 文字は省略して書き、a[5]には自動的に'¥0'が代入される。また、これらは配列の初期化のルールに従うので、配列の要素数を省略した以下の形も使用可能である。

```
char a[] = {'A', 'k', 'i', 't', 'a', '¥0'};
char a[] = "Akita";
```

これらは配列の要素数が自動的にカウントされることになり、この場合では要素数が 6 になる。
注意点として、次に示す間違った使用例がある。

```
char a[10];
a = "Akita";
```

この場合では、a は配列の先頭アドレスを取り扱っているので、コンパイルエラーとなる。
次に、文字列を操作するライブラリを用いて、配列に文字を代入する方法を紹介する。具体的には、次のようにする。

```
        strcpy(a, "Akita");
```

これで配列に文字が格納できる。この方法を使う場合はプログラムの最初に#include

<string.h>とライブラリをインクルードする必要がある。また、stdio.h のインクルードだけで良い方法もある。

```
sprintf(a, "Akita");
```

printf 関数とよく似ており、書式指定することでもう少し複雑な処理も可能であるが、詳しい説明は省略する。文字型配列のイメージは、図２のようになる。

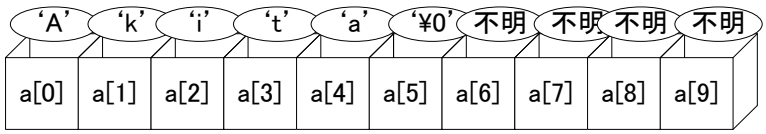

図２　文字型の配列 a[10]を用意して、それに"Akita"を格納。一つの箱には、一つの英数字しか入れられない。そして、最後に¥0 が入る。

配列に格納された文字列を表示するプログラムは次のようになる。

プログラム例４

```
#include <stdio.h>

int main(void)
{
        char a[10];             /* 文字型の変数 */

        sprintf(a, "Akita" );   /* 代入 */
        printf( "%s¥n", a );    /* 表示 */

        return 0;
}
```

出力

```
Akita
```

※注意

配列を使う場合、そのサイズはプログラマが決めなくてはならない。そのサイズを超えて代入するような操作をすると、エラーメッセージを出して止まる。通常は十分大きいサイズで使う。

日本語の場合

文字型配列の使い方

文字型の場合、それに格納できるデータは 1 バイトである。一方、日本語の場合、文字は 2 バイトで表現する。そのため、日本語は 2 個の文字型のデータ領域に 1 個の文字を格納している。文字型の配列に日本語の文字を格納する方法は、アルファベットの場合とほとんど同じである。従って、以下のようになる。

```
strcpy(a, "秋田");
sprintf(a, "秋田");
```

ただし、a[0]='秋';のような代入は正しく動作しない。a[0]は 1 バイトで、日本語の「秋」は 2 バイトだからである。文字型配列のイメージは、図 3 のようになる。

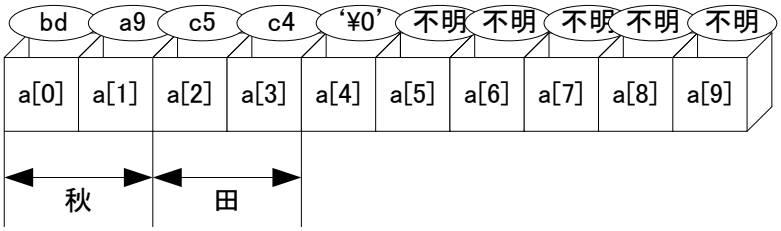

図３　文字型の配列 a[10]を用意して、それに"秋田"を格納。日本語の場合、 2 つの箱で 1 つの文字が入る。そして、最後に'¥0'が入る。

表示方法もアルファベットと同じである。

プログラム例５

```
#include <stdio.h>

int main(void)
{
        char a[10];

        sprintf(a, "秋田");
        printf("%s¥n", a);

        return 0;
}
```

出力

```
秋田
```

※注意

配列のサイズに気を付ける必要がある。日本語はアルファベットとは異なり、サイズは 2 倍となる。さらに、文字列の終わりを示す'¥0'を加えた配列の要素数が必要である。

文字列のキーボードからの入力

文字列をキーボードから入力する方法を示す。まずは、使い慣れている scanf である。

プログラム例６

```
#include <stdio.h>

int main(void)
{
        char a[100];

        scanf("%s", a);
        printf("%s¥n", a);

        return 0;
}
```

出力例(斜体は入力)

Akita
Akita

上記のように使用することができる。文字列をキーボードから入力し、Enter ボタンを押すと、'¥n'が'¥0'に置き換えられた上で、配列 a に格納される。ただし、注意が必要で、"This is a pen"と入力すると、最初の空白までが区切りとみなされてしまい、"This"までしか読み込まれない。1 行全

て入力する方法としては gets がある。

プログラム例７

```
#include <stdio.h>
int main(void)
{
        char a[100];

        gets(a);
        printf("%s¥n", a);

        return 0;
}
```

出力例(斜体は入力)

This is a pen
This is a pen

この方法だと、1 行全てが入力され、また、'¥n'は'¥0'に置き換えられる。ただし、この方法でも注意が必要で、配列で確保された文字数以上を入力すると、問題が生じる可能性（メモリ破壊）があり、使用する場合には気をつけなくてはならない。そのため、次回で習う「fgets」で入力文字数を制限する方が良いが、今回は十分なサイズの配列でプログラムを組むことで対応する。

入力される空白の数があらかじめ分かっている場合には以下のように scanf を利用することもできる。

プログラム例８

```
#include <stdio.h>
int main(void)
{
        char a[10], b[10], c[10], d[10];

        scanf("%s %s %s %s", a, b, c, d);
        printf("%s %s %s %s¥n", a, b, c, d);

        return 0;
}
```

出力例(斜体は入力)

This is a pen
This is a pen

演習
　プログラム例 1～例 7 を作成し、実行しなさい。

課題 2
Kadai2-1　以下のプログラムを作成、実行し、結果について考察しなさい。以下、考察が求められた場合はソースプログラム、動作結果、考察を提出すること。

```
#include <stdio.h>

int main(void)
{
        char a[] = {'T', 'A', 'K', 'E'};

        printf("%s¥n", a);

        return 0;
}
```

Kadai2-2　以下のプログラムを作成、実行しなさい。また、gets を scanf に変えたものを作成し、その際に起こる問題を考察しなさい。

```
#include <stdio.h>

int str_length(char a[]);

int main(void)
{
        char str[100];
        int length;

        printf("英文を入力して下さい:");
        gets(str);

        length = str_length(str);

        printf("文字列の長さは%dです¥n", length);

        return 0;
}

int str_length(char a[])
{
        int i;

        for(i = 0; a[i] != '¥0'; i++)     // '¥0'が出現するまでiをカウントアップ
                ;                         // 何もしない

        return i;                         // for文を抜けた後のiが文字列の長さになる
}
```

課題２は続きがあるので、締切は後で提示する。