14. 文字列

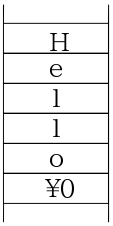
C 言語には文字列型といったものはない。そこで、"Hello"といった文字列を扱うには、文字が並んだもの、すなわち文字の配列と考える。

文字列の入出力

キーボードから文字列を入力するには、

```
char st[20];              /*   宣言   */
scanf("%s", st, sizeof(st));   /* 配列名だけを書くときは & をつけない */
                               /* st で配列の先頭のアドレスになる */
```

のようにする[1]。
ここで、キーボードで Hello と叩くと



のように配列に格納される。ここで、最後に入っている \0 はヌルコードと呼ばれる文字列の終りを表す記号 (ASCII コードは 0) である。C 言語では、文字列の最後は必ず \0 にすることにしているので、配列宣言時にはこの最後の \0 も入るような大きさにしなければならない。
他に、scanf ではなく getchar() を使って一文字ずつ読み込む方法もあるが、その場合には 次のように \0 を自分で書き込む必要がある。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    int  ch;
 5    char st[100];
 6    int  i = 0;                          /* 配列 st のインデックス用 */
 7
 8    while((ch = getchar()) != ' ') {     /* 入力文字列がスペースで終っている場合 */
 9      st[i] = ch;
10      i++;
11    }
12    st[i] = '\0';                        /* 文字列の最後には '\0' を入れる */
13  }
```

[1] scanf("%s",&st) ではなく、scanf("%s",st) となっているのは、scanf が変数領域へのポインタを要求しているからである。しかしながら、

```
char *str;
scanf("%s", str); /* まちがい!  */
```

としてしまうと、値を確保する領域が取られていないのでエラーとなる。
また、一般の C 言語処理系は scanf("%s", st); でよいのだが、VisualStudio では scanf_s("%s", st, sizeof(st)); のように領域のサイズを指定する必要がある。

文字列を出力する場合は、次のようにする[2]。

```
1  #include <stdio.h>
2  main()
3  {
4       char st[20];                /*   宣言   */
5
6       scanf("%s", st);            /* 配列名だけを書くときは & をつけない */
7
8       printf("%s", st);           /* st で配列の先頭のアドレスになる */
9  }
```

[例題 1] キーボードから Yamada_Taro のような名前 (姓と名の頭文字は大文字で、その間はアンダースコアのような記号を入れる) を入力し、そのイニシャルを求めるプログラムをつくりなさい。

```
1  /* 名前のイニシャル */
2  #include <stdio.h>
3  main()
4  {
5    int i=0;
6    char name[30];
7
8    scanf("%s", name);                  /* 配列変数名だけの時は & を付けない */
9    printf("%s's initial is ", name);
10
11   while(name[i]!='\0') {               /* 文字列の最後は \0 で終わっている */
12     if ('A' <= name[i] && name[i] <= 'Z')          /* 大文字ならイニシャル */
13       putchar(name[i]);
14     ┌──────┐
15     │      │;                            /* 配列のインデックスを 1 つ進める */
16     └──────┘
17   }
18 }
```

[例題 2] キーボードから名前、年齢、電話番号 (例えば、Yamada_Taro 20 045-563-1141) を入力し、適当な変数にその値を代入するプログラムをつくりなさい。

```
1  /* 名前・年齢・電話番号 */
2  #include <stdio.h>
3  main()
4  {
5    char name[30];
6    int  age;
7    char tel[20];
```

[2] printf("%s", st); のように st[0] といった配列要素を指定しないのは、%s の書式がポインタを受け取り \0 に出会うまで出力するという仕様になっているからである。

```
 8
 9    scanf("%s", name);              /* 配列変数名だけの時は & を付けない */
10    scanf("%d", &age);              /* 普通の変数名の時は & を付ける */
11           ┌──────┐
12    scanf( │      │ );             /* tel の読み込み */
13           └──────┘
14    printf("%s  %d  %s\n", name, age, tel);
15  }
```

[例題 3] 前の例題で、複数の人のデータを入力できるように変えなさい。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    int i;
 5    char name[3][30];              /* 三人分の領域を確保（二次元の配列）*/
 6    int  age[3];
 7                ┌──────┐
 8    char tel    │      │ ;
 9                └──────┘
10    for(i=0; i<=2; i++) {
11      scanf("%s%d%s", name[i], &age[i], tel[i]);  /* 3 つの scanf を書くかわり */
12    }
13    for(i=0; i<=2; i++) {
14      printf("%-20s %3d %-14s\n", name[i], age[i], tel[i]);
15    }
16  }
```

(注 1) 上の printf の中で %-20s などとしているのは、20 文字分のスペースをとってそこに左づめにして出力するということ。

(注 2)char name[3][30]; は 30 文字が入る配列が 3 つということで、3 行 30 列の領域に

	0	1	2	3	4	5	6	7	8	9	…	28	29
0	S	a	t	o	\0								
1	Y	a	m	a	d	a	\0						
2	K	i	d	o	\0								

のように文字が格納される。このような 2 次元配列に文字列を順々に格納していくには、
scanf("%s", name[0]); のようにどの行に名前を読み込むかまでを指定する。
同様に printf("%s", name[0]); についても、行のみの指定だけで \0 が見つかるまで出力される。

文字列の代入

char 型を指すポインタを宣言し、文字列 (の先頭の番地) を代入する。

```
#include <stdio.h>
main()
{
     char *str;                  /*   宣言    */

     str = "abcde";              /* 文字列を定義し、先頭のアドレスを代入する */
     printf("string = %s\n", str);
}
```

文字列の初期化

宣言時に文字列 OK を定義するには、

```
     char str[ ] = "OK";                        /* 定義 A */
```

または

```
     char *str = "OK";                          /* 定義 B */
```

のようにする。定義 A の場合、配列要素数を書かなくてよい。コンピュータがその文字列を格納するのに十分な要素数を自動的に計算してくれる。この例の場合、一番最後に \0 を入れるので要素数は 3 である。定義 A と定義 B では内部形式が異なるので次のような注意が必要である。定義 A では、文字列中の個々の文字を陽に配列として格納しているので、文字列中の一部の文字を入れ換えるといった破壊的な代入も可能となる。一方、定義 B ではシステムによって文字列が静的な領域にとられ、その先頭アドレスが str に代入されるので、文字列中の一部を入れ換えようとするとエラーとなる。もちろん値を参照するだけなら、文字列中の一部だけをアクセスしても全く問題ない。また、定義 B 後に

```
     str = "12345";
```

のようにポインタを入れ換えることも問題ない。(str はポインタ**変数**です!)

例題 2 の 3 人の名前を宣言時の初期値として代入するのに、次のようにすると前もって 20 文字分の領域を確保する必要がなくなる。

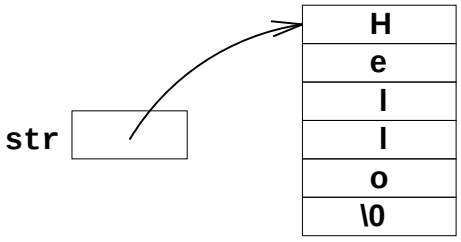
```
1  #include <stdio.h>
2  main()
3  {
4    char *name[3]={"Sato","Yamada","Kido"};
5    int i;
6
7    for(i=0; i<=2; i++)
8      printf("name[%1d] = %s\n", i, name[i]);
9  }
```

文字列とポインタ変数

"Hello"という文字列[3]を変数に代入したい時は、

```
char *str;          /* 宣言 */
str = "Hello";
```

のようにした。ここで、char *str; という宣言は変数 str が char 型データを参照するためのポインタ変数であることを示している。図示すると下のようになる。



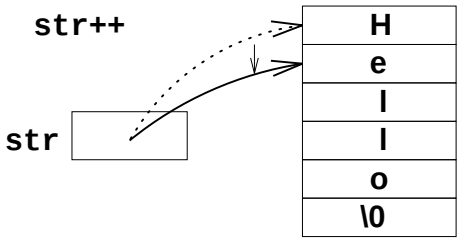
ここで、最後に入っている \0 はヌルコードと呼ばれる文字列の終りを表す記号である。
ポインタというのは番地を格納する変数である。そしてその番地には何が入っているかを示すために char といった型を宣言時につける。実際の番地の値を印刷しても意味がないので、図上では矢印を使って表現する。指し示している実体を見るには、

```
if (*str != 'A')
        putchar(*str);
```

のように変数の前に 間接演算子 * をつけて *str とする。上の文を実行すると、*str は 'H' なので H が出力される。ここで

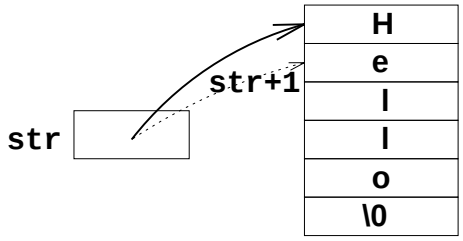
```
str++;
```

を実行すると、ポインタが 1 つ進んで e を指すようになるので (下図)、この状態で再度上の if 文を実行すると e が出力される。



ポインタ str の値を変えずに *(str+1) としても 2 番目の要素 'e' をアクセスできる。

[3]C 言語では、文字は一重引用符、文字列は二重引用符で囲む。



[例題 4] 2 つの文字列が同じものかどうかを判定するプログラムを作りなさい。

[問題に対する 1 つの考え方]
[1] 2 つの文字列 str1、str2 がそれぞれ配列に格納されているものとする。
[2] str1 の終りまで、配列の最初の要素から一文字ずつ同じかどうか比較していく。ただし、同じでなかったらただちにこの作業を止める。
[3][2] の作業が終った段階で、str1、str2 ともに文字列の最後の要素を比較していたなら、2 つの文字列は同一のものである。そうでなければ、2 つは同一でない。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    int i = 0;
 5    char  str1[ ] = "Hello";
 6    char  str2[ ] = "Hello!";
 7
 8    while (str1[i] != '\0') {
 9      if (str1[i] != str2[i])              /* 違うことが分かったら    */
10        break;                             /* 直ちにループから抜ける */
11      ┌──────┐
12      │      │;                           /* 配列のインデックスを 1 つ進める */
13      └──────┘
14    }
15    if (str1[i]=='\0' && str2[i]=='\0')
16      printf("%s and %s are identical strings.\n", str1, str2);
17    else
18      printf("%s and %s are not identical strings.\n", str1, str2);
19  }
```

break 文について
手順 [2] の中の「強制的に繰り返しループから抜ける」ことを C 言語で実現するには上のように break 文を使う。上の例で言えば、8 行目の if 文の条件が成り立ち、9 行目の break 文が実行されると、while 文のループから抜けて処理は 14 行目に移る。
for 文や while 文が何重にも入れ子になっている場合には、その break 文を含む最も内側のループから脱出できるだけで、すべてのループから一度に抜けることはできない。

例題 4 のプログラムを配列の表現ではなく、ポインタを用いて書き換えると次のようになる。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    char  *str1 = "Hello";
 5    char  *str2 = "Hello!";
 6    char  *p1, *p2;
 7
 8    p1=str1;                                  /* 先頭のアドレスを保存 */
 9    p2=str2;
10
11    while (*str1 != '\0') {
12      if (*str1 != *str2)                     /* 違うことが分かったら    */
13        break;                                /* 直ちにループから抜ける */
14      str1++;                                 /* ポインタを 1 つ進める */
15      str2++;
16    }
17
18    if (*str1=='\0' && *str2=='\0')
19      printf("%s and %s are identical strings.\n", p1, p2);
20    else
21      printf("%s and %s are not identical strings.\n", p1, p2);
22  }
```

[例題 5] 文字列の文字を 1 字ずつ取り出して出力するプログラムを作りなさい。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    char *str;
 5
 6    str = "How are you?";
 7    while(*str != '\0') {          /* 文字列の最後は'\0'    授業で使っている */
 8      putchar(*str);               /* コンピュータでは、\は円記号 */
 9      ┌──────┐
10      │      │;                   /* ポインタが次の文字を指すようにする */
11      └──────┘
12    }
13  }
```

[例題 6] 上の例題で、小文字は大文字に、大文字は小文字に変換して出力するように直しなさい。

[例題 7] 文字列の長さ (\0 は含まない) を求めるプログラムを作りなさい。

```
 1  #include <stdio.h>
 2  main()
 3  {
 4    char *pt, *str = "How are you?";   /* str には文字列の先頭番地を代入 */
 5    int i=0;                           /* 長さを格納するカウンタ、初期値は 0 */
 6
 7    pt = str;                          /* 先頭のアドレスを保存 */
 8    while(*str != '\0') {              /* 文字列は'\0' で終る */
 9      i++;
10      ┌──────┐
11      │      │;                       /* ポインタが次の文字を指すようにする */
12      └──────┘
13    }
14    printf("length of \"%s\" = %d\n", pt, i);  /* "を出力するには \" とする */
15  }
```

[例題 8] 上のプログラムでは '\0' が見つかるまでポインタを進めてしまっているので、6 行目で最初の str の値を別のポインタ変数 pt で保持しておかないと 13 行目の文字列の表示で困ることになる。ポインタ str の値を変えずに str+i という形を用いれば pt は不要になる。この考え方で上のプログラムを書き変えなさい。

文字列処理用の組込関数

C 言語で文字列処理を行う際には、<string.h> というヘッダーファイルをインクルードし、以下のような組込関数を使うのがよい。

関数	意味
strcpy(s1, s2)	s2 を s1 にコピーする
strcat(s1, s2)	s1 のあとに s2 を連結する
strcmp(s1, s2)	s1 と s2 が同じ文字列なら 0 を返し、異なる場合は 0 以外の値となる
strlen(s)	文字列 s の長さを求める

配列とポインタ

C 言語では、配列とポインタは密接なつながりがある。

```
char a[5];
```

と宣言された配列において、配列名 a は配列の先頭の要素の番地である (定義) ので、a[0] と書く代わりに *a としてよい。同様に、a[1] は *(a+1) としてよい。すなわち、

$$
\begin{array}{|l|}
\hline
a[0] \equiv *a \\
a[1] \equiv *(a+1) \\
a[2] \equiv *(a+2) \\
a[3] \equiv *(a+3) \\
a[4] \equiv *(a+4) \\
\hline
\end{array}
$$

しかし、配列とポインタを混同してはいけない。例えば

```
char st[10];          /* 宣言 */
st = "123";           /* 不可 */
```

は、エラーとなる。(st は st[10] の宣言で、領域の先頭を指す番地となっており、それを変えようとしているため。)　また、配列名は変数ではないので、

```
char st[10];          /* 宣言 */
st++;                 /* 不可 */
```

といったこともできない。