配列と文字列

文字の列を " " で囲ったものを文字列リテラルと呼んだ。

文字列リテラル "Hello!\n" の表示　　printf("Hello!\n");

文字列は文字の配列として表現される。ただし末尾は空文字（ヌル文字・ヌルコード）'\0' で終わる。

文字列リテラル "abc" は char 型の配列としてメモリ上に配置される。

'a'	'b'	'c'	'\0'

'\0' は文字列の終わりを示す印（空文字）

文字列リテラル "A" は二つの文字定数 'A' と '\0' からなる文字型の配列。
文字定数 'A' は A という 1 文字。

文字列と配列

```
char data[100];

data[0] = 'H';
data[1] = 'e';
data[2] = 'l';
data[3] = 'l';
data[4] = 'o';
data[5] = '!';
data[6] = '\0';
printf("%s\n", data);
```

H	e	l	l	o	!	\0	

配列 data は、文字列 "Hello!" を表している

空文字以前の文字までの要素数を
文字列の長さという。

この例の場合、文字列の長さは 6

配列名（ポインタ）

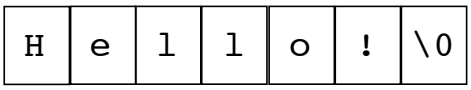
%s は、文字列を表示する変換指定

配列 data の要素数は 100 で宣言してあるので、添え字が 99 を越える代入は範囲外参照になる（data[100]='\0' としてはいけない）。

文字配列の初期化と表示

```
char string[] = "Hello!";
int i=0;

printf("%s", string);

while( string[i] != '\0' ){
  printf("%c", string[i]);
  i++;
}
```

←―― 文字型の配列 string を



として初期化

配列の大きさは自動的に、文字数 + 1 になる(この場合は 7)

`printf("%s", string)` 変換指定 %s を用いて、文字配列 string を表示。

`while( ... )`

while ループを用いて、文字配列要素を 1 つずつ出力。つまり、空文字 '\0' に至るまで、添え字 i をインクリメントしながら、配列要素 string[i] を文字として表示。

文字列のサイズ

"Hello!" という文字列は、文字型の配列としてメモリ上に格納される。

```
char string[]="Hello!";

printf("Size of %s is %d\n", string, sizeof(string));
```

```
char string[20];

string[0] = 'H';
string[1] = 'e';
string[2] = 'l';
string[3] = 'l';
string[4] = 'o';
string[5] = '!';
string[6] = '\0';

printf("Size of %s is %d\n", string, sizeof(string));
```

文字列の読み込み 1

getchar を用いて 1 文字ずつ文字を文字配列に読み込む。
読み終わったら、文字配列の最後の要素に空文字を代入。

```
char string[100];
int c, i=0;

while( (c=getchar()) != EOF ){
  string[i] = c;
  i++;
}
string[i] = '\0';

printf("%s\n", string);
```

この例では文字を格納する配列 string の要素数は 100 で宣言してあるので、99 文字以上の入力があると範囲外参照となる。
添え字の正しい範囲はプログラマの責任。

文字列の読み込み 2

scanf を用いて文字列を文字配列に読み込む。

```
scanf("%s", 配列名)
```

キーボードから空白文字で区切られた文字列を読み込み、配列名のアドレス（配列）に格納。配列名の前に & を付けない！

scanf 入力で変換指定 %s を使うと、入力された文字列は、配列名で指定されるアドレスから始まるメモリ領域に格納される。

```
char string[100];

scanf("%s", string);

printf("%s\n", string);
```

string の前にアドレス演算子 & は不要！

この例でも 99 文字以上の入力があると配列の範囲外参照となる。

文字列の出力

```
char string[] = "How are you?";
int i;

printf(“%s\n”, string);

i=0;
while( string[i] != '\0' ){
  putchar( string[i] );
  i++;
}

puts(string);
```

1) printf() で %s を用いる

2) putchar() で 1 文字ずつ出力

3) puts() を用いる

puts 関数は引数として文字列リテラルを受け取り、その内容を出力。
出力後改行する。

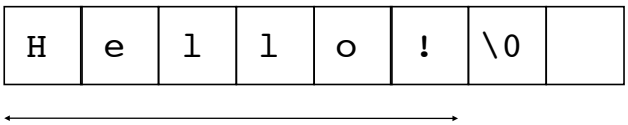
```
printf("Hello!\n");

puts("Hello!");
```

同じ結果

文字列の操作 1

文字列の長さを数えるには、空文字が出現するまでの要素数を数えれば良い。

文字列は文字型の配列
最後に空文字がある。



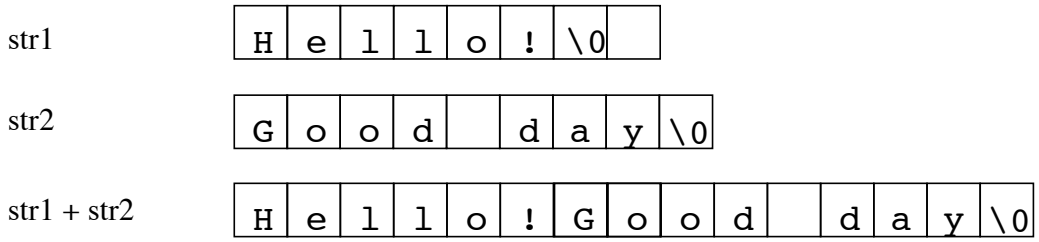
文字列の長さ(この場合は 6)

```
char string[] = "Beckham visited Japan!";
int len = 0;

while( string[len] != '\0' )
  len++;

printf("文字列 %s の長さは %d です。\n", string, len);
```

while ループにより空文字に出会うまで添え字をインクリメント。

文字列の操作 2

文字列を連結したい(2 つの文字列をつなげる)。最初の文字列の空文字から連結する文字列の内容をコピーする。




```
char str1[100] = "Beckham visited Japan!";
char str2[] = "With his wife Victoria";
int i=0, j=0;

while( str1[i] != '\0' ) i++;

while( str2[j] != '\0' )
  str1[i++] = str2[j++];
str1[i] = '\0';
```

インクリメント演算子再考

i ++ の評価値は、i の値を1 増やす前の値。

int i = 0;

printf("%d\n", i++); ⟶ 表示は 0
printf("%d\n", i++); ⟶ 表示は 1

文字列の操作 3

```
#include <string.h>
```

ヘッダファイル string.h をインクルードすることで、文字列操作関連のライブラリ関数が使用できる。

文字列の長さを得る関数 strlen()

```
strlen( string )
```

引数として文字列配列名 string を受け取り、文字列の長さ(int) を返す関数。文字列終端の空文字は数えない。

関数へ渡す値のことを引数という。

```
char string[] = "Hello!";
int len;

len = strlen(string);
printf("%d\n", len);
```

実行結果は、6 となる。

文字列の操作 4

2 つの文字列を連結する関数：strcat()

文字列 string1 の後に文字列 string2 を連結する。

```
strcat( string1, string2 )
```

string1 の終端の空文字を取り除き、その後ろに string2 を連結。

文字列 string1 は連結により長くなるので、あらかじめ十分大きな領域を確保しておく必要がある（下の例では 100 文字分の領域を確保）。

```
char str1[100] = "Hello!";
char str2[] = "How are you?";

strcat(str1, str2);
printf("%s\n", str1);
```

実行結果は、Hello!How are you? となる。

文字列の操作 5

文字列をコピーする関数：strcpy()

文字列 string1 に文字列 string2 をコピーする。

```
strcpy( string1, string2 )
```

string1 の長さは、コピーされる文字列 string2 を格納できる大きさでなければならない（下の例では 100 文字分の領域を確保）。

文字列の代入演算子に相当する。

```
char str1[100] = "Hello!";
char str2[] = "How are you?";

printf("%s\n",str1);
strcpy(str1, str2);
printf("%s\n", str1);
```

strcpy により文字列 str1 は str2 によって上書きされる。

問題 1

scanf() を用いて 2 つの文字列を読み込み、次の文字列操作を行うプログラム。

1) str1, str2 の長さを表示。
2) str1 と str2 を連結した文字列を表示、その長さも表示。
3) str1 に str2 をコピーした結果を表示、その長さも表示。

文字列操作の関数を使わないプログラムと、使うプログラムの 2 通りつくれ。

```
% ./a.out
第 1 文字列を入力:Brazil
第 2 文字列を入力:Germany
第 1 文字列 Brazil の長さは 6
第 2 文字列 Germany の長さは 7
連結した文字列は BrazilGermany
コピーした結果は Germany
%
```

この色はプログラムによる出力。

問題 2

自然数を 10 進数表記で入力し、これを 2 進数表記で出力するプログラム

```
% ./a.out
自然数を 10 進数で入力: 25
2 進数表記は: 11001
%
```

この色はプログラムによる出力。

ヒント

```
25 / 2 = 12 --- 1
12 / 2 =  6 --- 0
 6 / 2 =  3 --- 0
 3 / 2 =  1 --- 1
 1 / 2 =  0 --- 1
```


1) データを整数値として入力
2) 2 で割った商が 0 になるまで割り続ける

余りを逆順で書いたものが 2 進数表記。
余りを配列として格納しておいて、最後に逆順で出力。

25 = 1*16 + 1*8 + 0*4 + 0*2 + 1*1

問題 3

自然数を 10 進数表記で入力し、これを 16 進数表記で出力するプログラム
変換指定 %X を使ったものと比較すること

```
% ./a.out
自然数を 10 進数で入力: 25
16 進数表記は: 19
% ./a.out
自然数を 10 進数で入力: 90
16 進数表記は: 5A
```

この色はプログラムによる出力。

```
printf("%X\n", data);
```

整数値 data を16 進数で出力

ヒント　25 = 1*16 + 9*1 だから 16 進数表記は 19

90 = 5*16 + 10*1 だから 16 進数表記は 5A

問題 4

自然数を 16 進数表記で入力し、これを 10 進数表記に変換するプログラム
16 進数の入力チェックを行うこと。'0' ~ 'F' 以外の入力はエラー。

```
% ./a.out
16 進数表記で入力:41
16 進数表記の 41 は 10 進数表記では 65
% ./a.out
16 進数表記で入力:5A
16 進数表記の 5A は 10 進数表記では 90
% ./a.out
16 進数表記で入力:3G
入力エラー
%
```

16 進数表記は '0', '1', ~ 'F' の 16 個の記号を用いるので入力値は数値ではなく文字として取り扱う。getchar() を用いる。

'0' ~ 'F' 以外の入力はエラー。

入力チェック後、文字型の配列に 16 進数表記の数を格納する。桁数を記憶させる必要が有る。
その後、10 進数表記に変換して表示。

char data[10]; /* 10 桁までの 16 進数を取り扱う */

16 進数の 5A は 10 進数表記では、10 (A) + 5*16 = 90 である。

data[0] = '5', data[1] = 'A', len = 2 (入力は2桁)