プログラミング論I
(11) 文字列

電子情報工学専攻　日浦 慎作

ログインしておいてください

文字の表し方

- コンピュータは，数値しか扱うことが出来ない
- そこで，文字を扱うために，それぞれの文字に番号が振られている
  - 各文字に割り当てられた数値を文字コードと呼ぶ
  - 例えば，‘A’ は 65, ‘B’ は 66, ・・・
- この授業では，半角文字しか扱いません
  - 全角文字は　漢字コード　と呼ばれ，扱いが複雑.
  - 半角文字の規格は ASCII コードと呼ばれる

ASCIIコード表

出典 http://e-words.jp/p/r-ascii.html

文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進
NUL	0	00	DLE	16	10	SP	32	20	0	48	30	@	64	40	P	80	50	`	96	60	p	112	70
SOH	1	01	DC1	17	11	!	33	21	1	49	31	A	65	41	Q	81	51	a	97	61	q	113	71
STX	2	02	DC2	18	12	"	34	22	2	50	32	B	66	42	R	82	52	b	98	62	r	114	72
ETX	3	03	DC3	19	13	#	35	23	3	51	33	C	67	43	S	83	53	c	99	63	s	115	73
EOT	4	04	DC4	20	14	$	36	24	4	52	34	D	68	44	T	84	54	d	100	64	t	116	74
ENQ	5	05	NAK	21	15	%	37	25	5	53	35	E	69	45	U	85	55	e	101	65	u	117	75
ACK	6	06	SYN	22	16	&	38	26	6	54	36	F	70	46	V	86	56	f	102	66	v	118	76
BEL	7	07	ETB	23	17	'	39	27	7	55	37	G	71	47	W	87	57	g	103	67	w	119	77
BS	8	08	CAN	24	18	(	40	28	8	56	38	H	72	48	X	88	58	h	104	68	x	120	78
HT	9	09	EM	25	19	)	41	29	9	57	39	I	73	49	Y	89	59	i	105	69	y	121	79
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	J	74	4a	Z	90	5a	j	106	6a	z	122	7a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	K	75	4b	[	91	5b	k	107	6b	{	123	7b
FF*	12	0c	FS	28	1c	,	44	2c	<	60	3c	L	76	4c	\¥	92	5c	l	108	6c	|	124	7c
CR	13	0d	GS	29	1d	-	45	2d	=	61	3d	M	77	4d	]	93	5d	m	109	6d	}	125	7d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	^	94	5e	n	110	6e	~	126	7e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	_	95	5f	o	111	6f	DEL	127	7f

ASCIIコード表

スペース

終端記号

タブ文字

改行文字

数字

アルファベット
大文字

アルファベット
小文字

文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進
NUL	0	00	DLE	16	10	SP	32	20	0	48	30	@	64	40	P	80	50	`	96	60	p	112	70
SOH	1	01	DC1	17	11	!	33	21	1	49	31	A	65	41	Q	81	51	a	97	61	q	113	71
STX	2	02				"	34	22	2	50	32	B	66	42	R	82	52	b	98	62	r	114	72
ETX	3	03				#	35	23	3	51	33	C	67	43	S	83	53	c	99	63	s	115	73
EOT	4	04	DC4	20	14	$	36	24	4	52	34	D	68	44	T	84	54	d	100	64	t	116	74
ENQ	5	05	NAK	21	15	%	37	25	5	53	35	E	69	45	U	85	55	e	101	65	u	117	75
ACK	6	06				&	38	26	6	54	36	F	70	46	V	86	56	f	102	66	v	118	76
BEL	7	07				'	39	27	7	55	37	G	71	47	W	87	57	g	103	67	w	119	77
BS	8	08	CAN	24	18	(	40	28	8	56	38	H	72	48	X	88	58	h	104	68	x	120	78
HT	9	09	EM	25	19	)	41	29	9	57	39	I	73	49	Y	89	59	i	105	69	y	121	79
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	J	74	4a	Z	90	5a	j	106	6a	z	122	7a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	K	75	4b	[	91	5b	k	107	6b	{	123	7b
FF*	12	0c				,	44	2c	<	60	3c	L	76	4c	\¥	92	5c	l	108	6c	|	124	7c
CR	13	0d				-	45	2d	=	61	3d	M	77	4d	]	93	5d	m	109	6d	}	125	7d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	^	94	5e	n	110	6e	~	126	7e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	_	95	5f	o	111	6f	DEL	127	7f

C言語における定数の書き方

```
int i, j, k;

i = 65;
j = 0x41;
k = 'A';
```

この例では，i, j, k
すべてに同じ数値
(65)が代入される

10進数

- 普通に数値を書く
  - 整数の場合，最初の数字に0はダメ（8進数とみなされる）

16進数

- 0xに続けて16進数を書く
- 大文字でも小文字でも良い
  - 0x4E, 0x4e どちらでもよい

文字定数

- ‘ ’の中に基本1文字だけを書く

シングルクォート ‘ であることに注意！ダブルクォート ” ではない！

確かめてみよう！

```
#include <stdio.h>

int main(void) {
    int i;

    i = 'A';
    printf("A is %d¥n", i);

    return 0;
}
```

ex1.c

'A' の文字コードを表示するプログラム

実行結果

```
A is 65
```

- 整数の変数 i に 'A' の文字コードで ある65を代入している

- ということは

```
i = 'A';
```

とするのは

```
i = 65;
```

とするのと同じ.

文字を表示するには？

```
#include <stdio.h>

int main(void) {
    int i;

    i = 'A';
    printf("code of %c is %d¥n", i, i);
    return 0;
}
```

実行結果

```
code of A is 65
```

ex2.c

%d のかわりに %c を使う

- %d は，数値を10進数で表示する
- %c は，数値に対応した文字を表示する

試してみよう

```c
#include <stdio.h>

int main(void) {
    int i;

    i = 'A';
    printf("code of %c is %d¥n", i, i);
    return 0;
}
```

ここの行を
`i = 97;`
や
`i = 0x5a;`
などにしてみよう．どんな文字が出るか？

ex2.c

ASCIIコード表を見て，結果を予想してみよう

- 10進数の 97 は，どの文字か？
- 16進数の 5a は，どの文字か？

特殊文字（制御文字）

- 改行コードは，プログラム中に直接書くことができない
  かわりに ‘¥n’ を用いる

```
#include <stdio.h>

int main(void) {
    int i;

    i = '¥n';
    printf("=======%c=======", i);
    return 0;
}
```

ex3.c

実行結果

```
=======
=======
```

%c のところに，改行文字 ¥n が挿入されたものが表示されるので，実行結果は上のようになる

- 特殊文字は ¥ と，つづく1文字の組み合わせで表す．
- この場合のみ，’ と ’ の間に2文字書ける（文字コードは1つ）．
- 改行 ¥n のほかに，¥” ¥’ ¥¥ ¥t（タブ） などがある．

文字を表す変数型 char

- int は，１文字を表すには大きすぎる（メモリがもったいない）
  - int は通常，32bit の長さ．
  - ASCII文字コードには，１文字あたり 8bit あればよい．
- 文字を表すには char 型を用いる

```
char c = 'A';
printf("code of A is %c¥n", c);
```

- char型は8bitの整数型
  - ビット長が短いだけで，int型と同様に扱える
- 前ページまでのプログラム例は，変数 i の型をすべてchar型にしても問題なく動作する

文字のまとめ

- 文字には番号（文字コード）が振られている
- 各文字の番号を覚えていなくてもよい

```
i = 'x';
```

のように書けば，i に x の文字コードを代入できる

- これは，定数の書き方の1種である．

```
i = 120;
```
や
```
i = 0x78;
```
と同じ働き．

- 文字のための変数型 char が用意されている
  - 整数型なので，int と同じように使うことができる
  - ビット長が短すぎるので，通常の計算には用いられない

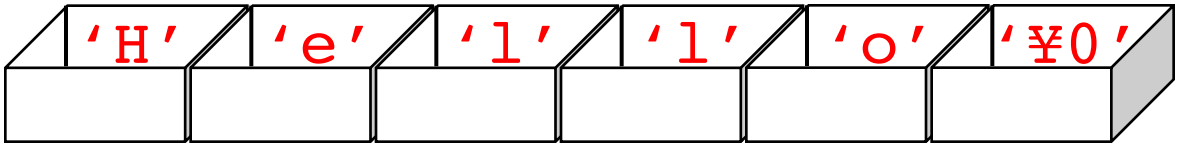
復習スライド

配列とは

- 数学でよく使う，添字付きの変数に対応

  $a_1, a_2, \cdots, a_N$ のように，変数を番号で指定する

- どういうときに便利なのか？
  - まとまった数のデータを処理するとき
    - クラスのテストの平均点を求めるなど
  - 変数に対して一括処理，繰り返し処理をしたいとき
    - すべての変数の値を2倍する，など
  - 文字列を扱うとき
    - 文字列（例：“Hello”）は文字の羅列なので，C言語では配列で取り扱う．来週の講義のテーマ．



文字列

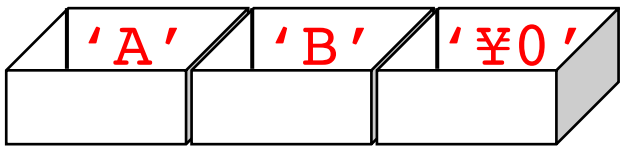
- char型の配列として表現する

```
char str[10];
```

str[0] が1文字目，str[1] が2文字目，・・・

- 文字列の定数は "ABC" のように表現
  - 文字列にはダブルクォート " " を用いる
- 【重要】文字列は，終わりに終端記号が置かれる
  - 終端記号は，'¥0' で表す（文字コードは 0 ）

```
char str[3] = "AB"; と書くのは，
str[0] = 'A';
str[1] = 'B';
str[2] = '¥0';
と同じ
```



文字列の初期化

- 配列を直接，文字列で初期化できる

```
char   s[] = “ABC”;
```

と

```
char   s[] = { ‘A’, ‘B’, ‘C’, ‘¥0’ };
```

は，同じ

- 【重要】要素数は，文字数＋１

  ‘¥0’ のぶんも確保する必要がある

  上の例は，以下と同じ

```
char   s[4] = “ABC”;
```

（配列の初期化を用いる場合，要素数を省略することが出来る）

確かめてみよう！

```c
#include <stdio.h>

int main(void) {
  char s[] = "ABC";
  int i;

  for(i = 0; i < 4; i++) {
    printf("s[%d] = %d, %c¥n", i, s[i], s[i]);
  }
  return 0;
}
```

ex4.c

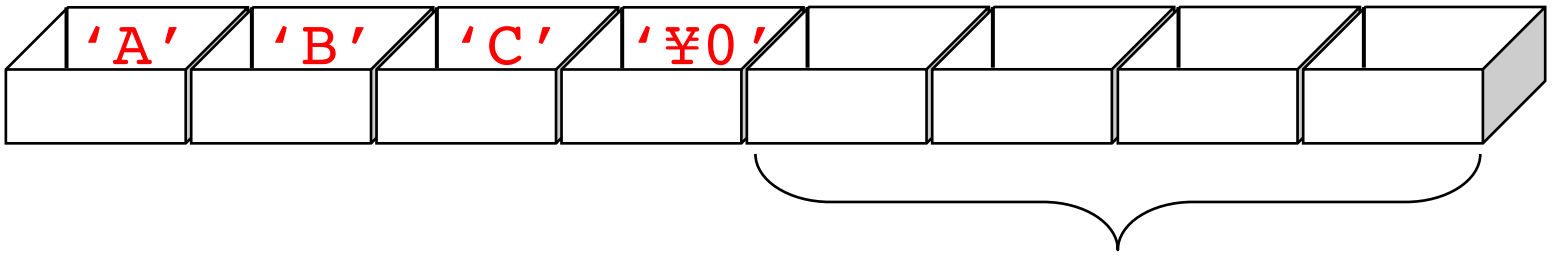
実行結果

```
s[0] = 65, A
s[1] = 66, B
s[2] = 67, C
s[3] = 0,
```

- 配列 s の要素数は4
- s の各要素には文字コードが入っている
- s[3] には終端記号の文字コードである 0 が入っている

終端記号(ターミネータ)の役目

```
char str[8] = "ABC";
```



- ターミネータが来るまでが文字列とする
  - ターミネータ以降は無視する
- 利点
  - 文字列の長さを別の変数に管理する必要がない
  - 同じ配列に，いろいろな長さの文字列を格納できる）

文字列を終わりまでたどる

```
#include <stdio.h>

int main(void) {
  char s[] = "ABC";
  int i;

  for(i = 0; s[i] != '¥0'; i++) {
    printf("s[%d] = %d, %c¥n", i, s[i], s[i]);
  }
  return 0;
}
```

実行結果

```
s[0] = 65, A
s[1] = 66, B
s[2] = 67, C
```

ex4b.c

- 2ページ前の「確かめてみよう！」と，ほぼ同じプログラム
- s[3] には終端記号の文字コードである 0 (='¥0') が入っている.
  - i == 3 のときは for 文に入らない

文字列の表示

- printf では %s で文字列を表示することが出来る
  この場合，文字列全体を渡すので，添字[ ]を取る

```
char str[] = "ABC";
printf ("string is %s¥n", str);
```

- 前述したように，%c は文字コードが表す
  1文字だけを表示するものなので，添字を付ける

```
printf ("character of 48 is %c¥n", 48);
printf ("character of str[0] is %c¥n", str[0]);
```

%s の s は string（一連のもの，ひも．転じて，文字列）を表す
%c の c は character（文字）を表す

プログラム例 Hello World! の表示

```
#include <stdio.h>

int main(void) {
  char s[] = "Hello World!";
  int i;

  printf("%s¥n", s);

  for(i = 0; s[i] != '¥0'; i++) {
    printf("%c", s[i]);
  }
  printf("¥n");

  return 0;
}
```

実行結果

```
Hello World!
Hello World!
```

ex5.c

文字列を，二通りの方法で表示

- 赤字：printfの文字列表示機能 %s を使った方法
- 青字：各文字を%cで，1文字ずつ表示する方法
  - 文字列の終わりを終端記号‘¥0’で判定している

`s[i] != ‘¥0‘` は，「文字列の終端記号に当たるまで繰り返し」を表す！

定番の書き方1

大文字・小文字の判定

- 大文字かどうか調べる方法（定番の書き方）

```
char c = 'M';
if('A' <= c && c <= 'Z') {
   //なにかの処理
}
```

- アルファベットのASCIIコードは，は，AからZまで，aからzまで，0から9まではそれぞれ連続していることを利用
  - 小文字かどうかの判定

```
if('a' <= c && c <= 'z')
```

  - 数字かどうかの判定

```
if('0' <= c && c <= '9')
```

定番の書き方2

大文字と小文字の変換

- 大文字を小文字に変換

```
char c = 'M';
if('A' <= c && c <= 'Z') {
   c = c – 'A' + 'a';
}
```

- 'A' を引くことで，何文字目かを求め，
  それに'a'を足すことで小文字に変換する
  - 大文字への変換

```
c = c – 'a' + 'A';
```

  - 数字の文字コードを，数値0~9に変換

```
i = c – '0';
```

定番の書き方3

文字列全体の処理

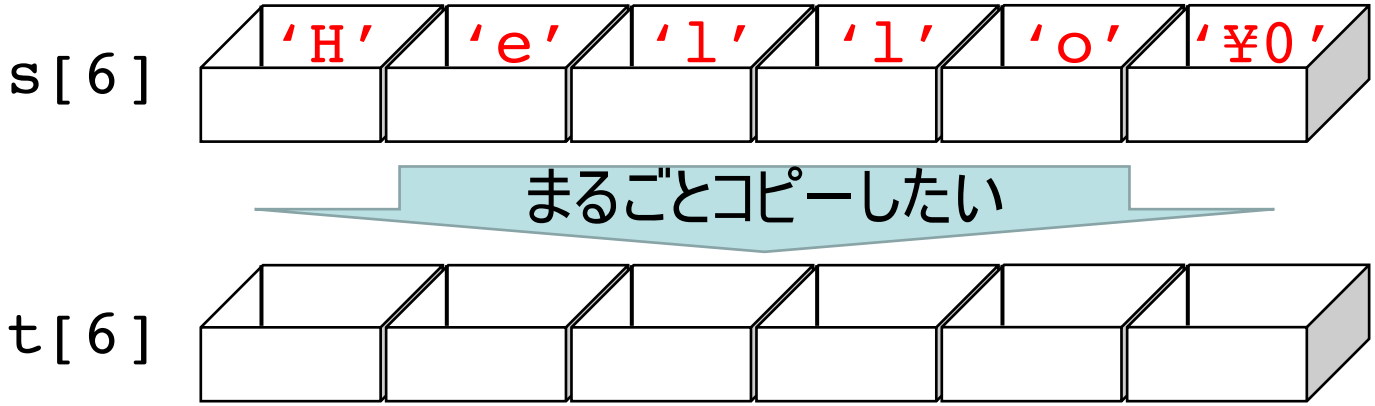
- 文字列全体について，１文字ずつ処理

```
char s[] = “Hello!”;
int i;

for(i = 0; s[i] != ‘¥0’; i++) {
   // なにかの処理
}
```

- i を 0 から増やしながら，sの要素を順に処理
  - s[i] が終端記号 ‘¥0’ に到達するまで処理
- メリット：文字列の長さを知らなくても処理できる

この書き方は，文字列のように終端記号を置いている場合しか使えません

文字列，配列のコピーについて




```
#include <stdio.h>

int main(void) {
  char s[6] = "Hello";
  char t[6];

  t = s;
  printf("%s¥n", t);

  return 0;
}
```

エラー

error.c

- 配列全体を，代入一発でコピーすることは出来ません
- 文字列も配列なので，文字列の代入は出来ません

方法1：for文で1個ずつコピー

方法2：文字列操作関数を用いる

```
#include <string.h>
strcpy(t, s);
```

文字列の読み込み

- scanfを使い，文字列を読み込む方法

```
char str[100];
scanf("%s", str);
```

十分大きなサイズにしておくこと！

- この場合，strに文字列を読み込む
  - 添字[i] はつけない
  - %s では & をつけなくてよい
  - スペースか，改行までの文字列を読み込んでくれる

なぜ %s のときだけ & をつけなくてよいかは，2年生で「ポインタ」を習うとわかりますが，
ここでは参考のため，簡単に説明しておきます．現段階では理解できなくても良いです．

- & は，変数のアドレス（メモリ上の場所，ポインタ）を求める演算子です．
  scanf は，そのアドレスへ，読み込んだデータを書き込みます．
- 配列 str[] では，添字の [] を省いて str と書くと，配列 str の先頭要素のアドレスを意味する，というC言語のルールがあり，それにより & を書かずにすみます．

文字(1文字)の読み込み

- scanfを使い，文字(1文字)を読み込む方法

```
char str[100];
scanf("%c", &str[i]);
```

- この場合，str[i]に文字(1文字)を読み込む
  - 添字[i] は，つけなければならない
  - %c では & をつけなければならない
  - つぎの1文字を読み込んでくれる
- 読み込み先の変数型は char型でなければならない

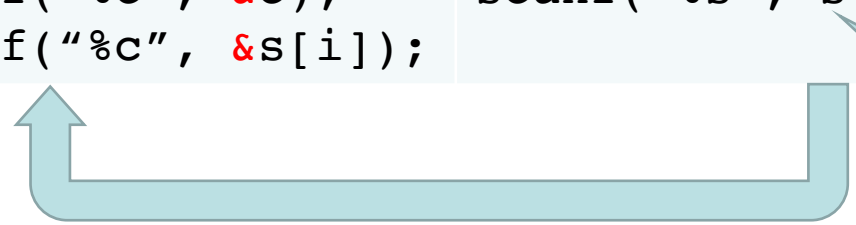
```
char c;
scanf("%c", &c);
```

文字と文字列　まとめ

	文字	文字列
変数	char c; のように char 型で表す	char s[5]; のように char 型の配列で表す
定数	'a' のように シングルクォートで囲む	"ABC" のように ダブルクォートで囲む　終端記号がつく

表示
(printf)
	%c で1文字を表示できる
	printf("%c", c);
	printf("%c", s[i]);
		%s で文字列全体を表示できる
		printf("%s", s);

入力
(scanf)
	%c で1文字を入力できる
	scanf("%c", &c);
	scanf("%c", &s[i]);
		%s で文字列全体を入力できる(&不要)
		scanf("%s", s);



十分大きなサイズにしておくこと！

文字列 s の要素1個 s[i] は，文字として扱う

文字列は，最後に終端記号が置かれる

プログラム例1

```
#include <stdio.h>

#define MAXLEN 100

int main(void) {
  char str[MAXLEN];
  int i, j;

  printf("Enter a string.¥n");
  scanf("%s", str);

  for(i = 0; str[i] != '¥0'; i++)
    ;

  printf("length of ¥"%s¥" is %d¥n", str, i);
  return 0;
}
```

- 入力された文字列の長さを調べるプログラム
- 文字列の途中に“ を置きたいときには，¥” と書く．

strlen.c

実行結果

```
Enter a string.
abc
length of "abc" is 3
```

for の後ろの ; は空行と呼び，繰り返し中に何もしないことを表す（iの値を進めることだけが目的）．

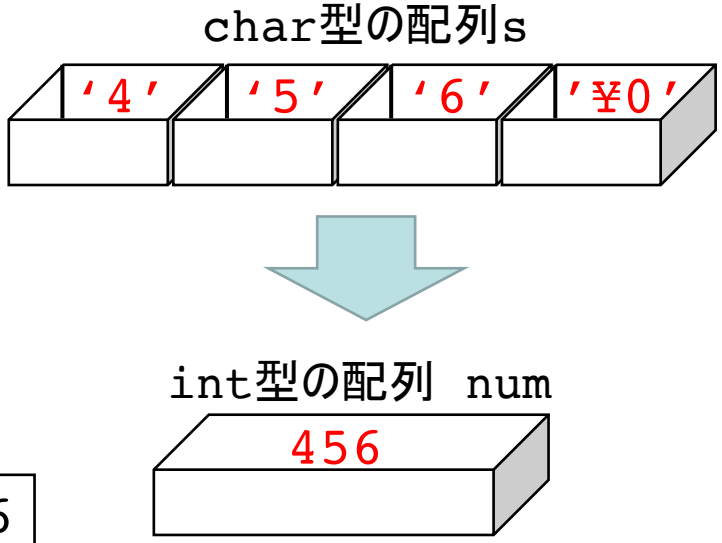
プログラム例2

```
#include <stdio.h>

int main(void) {
  char s[] = "456";
  int i, num;

  num = 0;
  for(i = 0; s[i] != '¥0'; i++) {
    num *= 10;
    num += s[i] - '0';
  }
  printf("%d¥n", num);

  return 0;
}
```

実行結果 456

atoi.c

- 数値を表す文字列（'0'～'9'だけを含む文字列）を，数値に変換するプログラム



プログラム例3

```
#include <stdio.h>

int main(void) {
  char str1[] = "Hello";
  char str2[] = "World";
  char str3[20];
  int i, j;

  for(i = 0; str1[i] != '¥0'; i++) {
    str3[i] = str1[i];
  }

  for(j = 0; str2[j] != '¥0'; j++) {
    str3[i] = str2[j];
    i++;
  }
  str3[i] = '¥0';

  printf("¥"%s¥" + ¥"%s¥" = ¥"%s¥"¥n", str1, str2, str3);
  return 0;
}
```

- 2つの文字列をつなげるプログラム
- 文字列の途中に“ を置きたいときには，¥” と書く.

実行結果

```
"Hello" + "World" = "HelloWorld"
```

concat.c