

プログラミング基礎

<http://bit.ly/prog2d>

数値表現 (2)

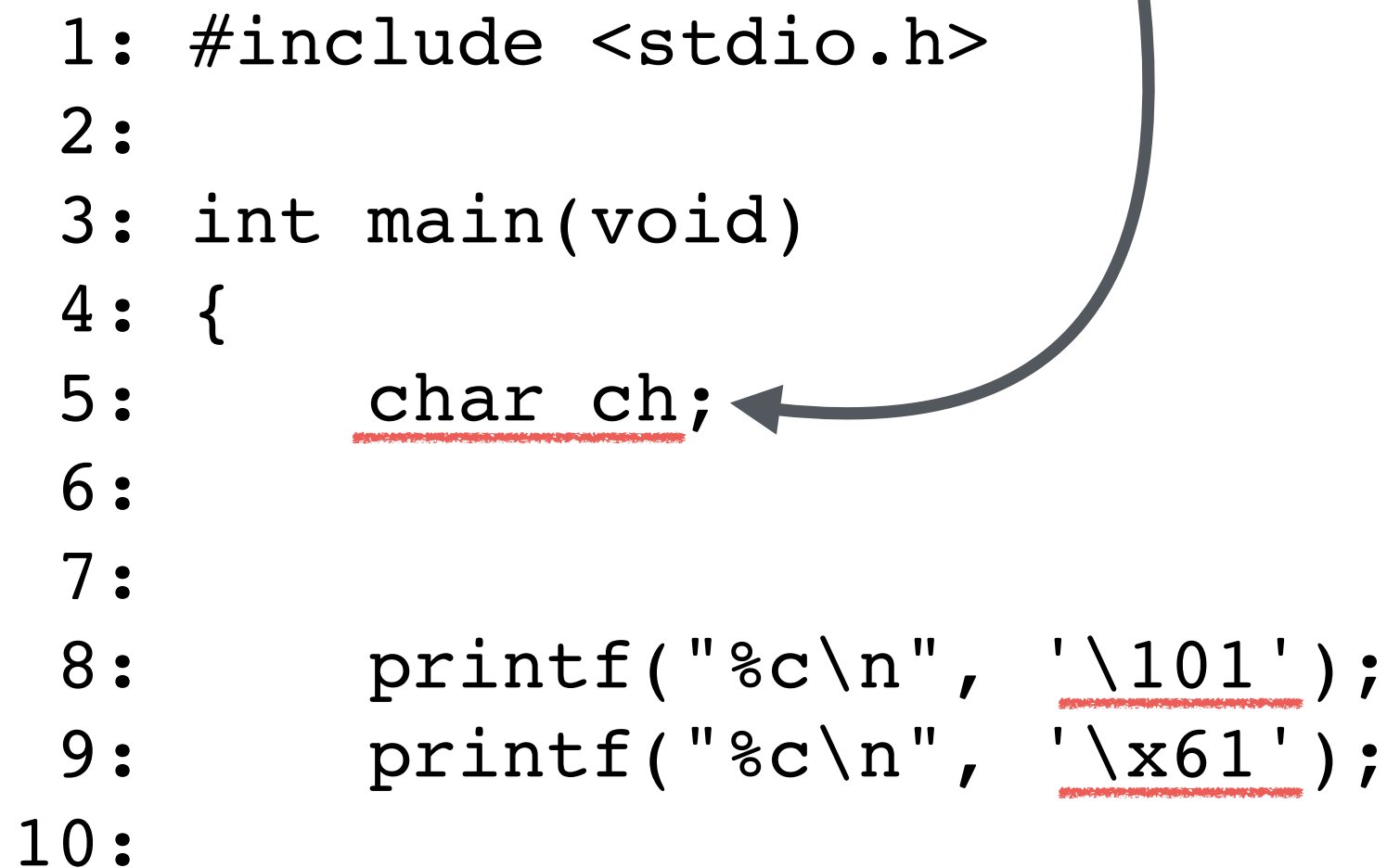
文字の表現について

前期 第12週

2017/7/10

【Point 1】コンピュータでの文字は、**数値に対応させ**、その数値(**文字コード**)で文字を表す。文字コードは、数値への対応の仕方や扱える文字の種類によって様々存在する。日本語を扱える文字コードには、JIS、Shift JIS、EUC、Unicode(UTF-8など)がある。 (p.33)

```
1: #include <stdio.h>
2:
3: int main(void)
4: {
5:     char ch;
6:
7:
8:     printf("%c\n", '\101');
9:     printf("%c\n", '\x61');
10:
```



【Point 3】文字は、「A」のように**シングルクォートで囲む**。
「\x41」 「\101」のように**16進数, 8進数**でも表すことができる。 (p.32 表2-1)

【Point 2】画面やプリンタに出力可能な英数字は、**ASCII**という文字コードによって、以下の表のように数値が割り当てられている（16進数表現の場合）。例えば、「F」の文字コードは「46₍₁₆₎または70₍₁₀₎」となる。

F の場合・・・ **上の桁** と **下の桁** を組み合わせて **46₍₁₆₎**

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_a	_b	_c	_d	_e	_f
2_		!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3_	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4_	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5_	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7_	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

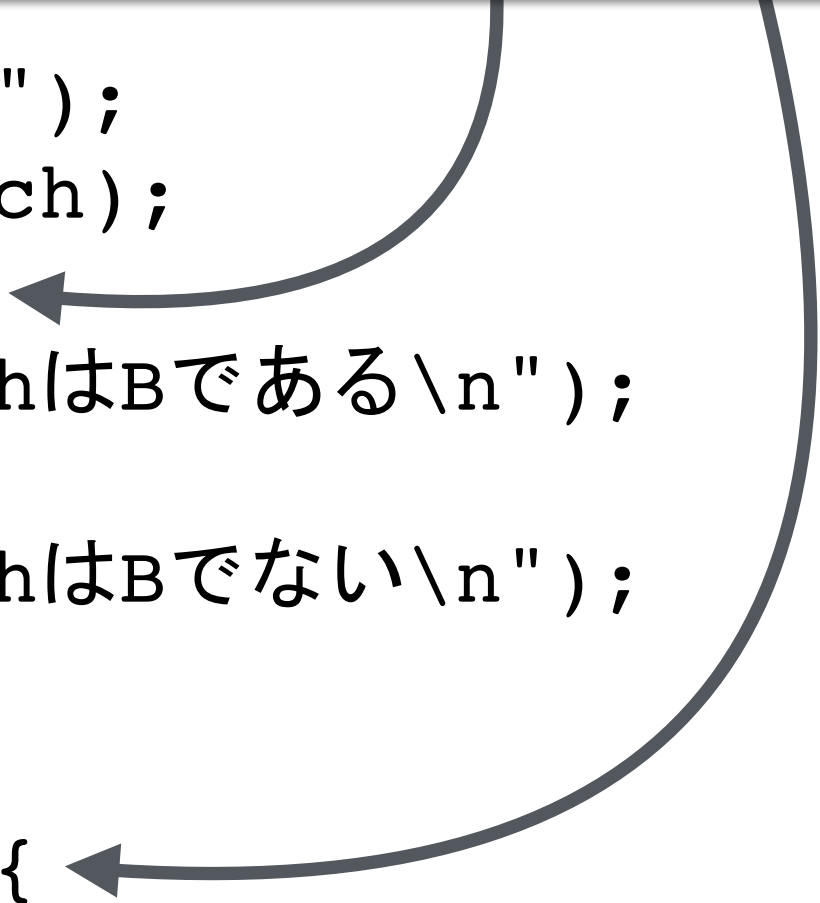
【Point 4】 文字は数値として処理されるため、char型とint型は必要に応じてキャストされる。

```
11:      ch = 'X';  
12:      printf("%c %d %x\n", ch, ch, ch);  
13:  
14:      printf("%c %c\n", ch+2, ch-1);  
15:
```

【Point 5】 char型は文字コードの整数としてint型と同様に加算/減算することができる。

【Point 6】 文字コードを比較することもできる。 (==, <, > など)

```
16:    printf("ch > ");
17:    scanf("%c", &ch);
18:    if(ch == 'B') { ←
19:        printf("chはBである\n");
20:    } else {
21:        printf("chはBでない\n");
22:    }
23:
24:    if(ch > 'Z') { ←
25:        printf("chはZより大きい\n");
26:    } else {
27:        printf("chはZより小さい\n");
28:    }
29:
30:    return 0;
31: }
```



【練習12-1】

**サンプルプログラムをコンパイルして、
実行結果を確認しましょう。**

【練習12-2】

サンプルプログラムの12行目で、文字コードを8進数でも出力するように変更して、「X」の文字コードを確認してみましょう。

(Xの文字コードは、8進数で表すと「130₍₈₎」)

【課題12-1】

16進数で入力された文字コードに対して、対応する文字を画面に出力する処理を繰り返すプログラムをmain()の中に作成してください。（int型の変数に対して「%x」の書式で入力し、出力する時は「%c」の書式を使うとできる。）ただし、次のような動作となるように作ってください。

- ▶ 入力された文字コードが「0（つまり整数のゼロ）」の場合は、その繰り返し処理を終了する
- ▶ 入力された文字コードが、出力可能な文字（つまり0x20～0x7eの範囲内）であれば、対応する文字を出力し、それ以外は「出力可能な文字ではない」ことを出力する

次ページの補足プログラムを参照してください

【課題12-1】

次のコードに必要な部分を追記して完成させましょう。

```
int main(void)
{
    int code;
    code = 1;          /* 0以外の初期値をcodeに代入する */
    while(code != 0) { /* codeが0になるまで入出力処理を繰り返す */
        printf("code > ");
        /* --- ここに、16進数で入力するscanfを追加する --- */
        /* 出力可能な文字かを調べて出力する */
        if(code >= 0x20 && code <= 0x7e) {
            /* --- ここに、codeの文字を出力するprintfを追加する --- */
        } else {
            printf("out of range\n");
        }
    }
    return 0;
}
```

【課題12-1】

[実行結果]

```
code > 23
```

```
code: #
```

```
code > 57
```

```
code: W
```

```
code > 84
```

(出力可能な文字の範囲外の場合)

```
out of range
```

```
code > 0
```

(0が入力されると繰り返しが終了する)

```
out of range
```

【課題12-2】

2つのchar型の引数に対して、大きい方の文字（つまり文字コードが大きい方）を返す関数max()を作成してください。

[この関数のプロトタイプ宣言]

```
char max(char c1, char c2);
```

```
/* c1とc2を比較して、大きい方の文字を返す */
```

【課題1 2-2】

[mainでの処理]

```
printf("max: %c\n", max('B', 'Z'));  
printf("max: %c\n", max('&', '='));
```

[実行結果]

```
max: Z  
max: =
```

【課題12-3】

2つのchar型の引数c1, c2に対して、その範囲に含まれる文字を、c1からc2まで順番に全て出力する関数show_chars()を作成してください。

[この関数のプロトタイプ宣言]

```
void show_chars(char c1, char c2);  
/* 文字コードとして、c1~c2の範囲に含まれる文字の出力を繰り返す */  
/*      ・ c1よりもc2の方が大きい場合、  
           c1からc2まで、文字コードを加算しながら出力する */  
/*      ・ c2よりもc1の方が大きい場合、  
           c1からc2まで、文字コードを減算しながら出力する */  
/*      (つまり、どちらの場合でも、c1から始まりc2で終わる) */  
/* 上記2つの場合に対応した繰り返し処理をそれぞれ作ればできあがる  
   (1つの繰り返し処理にまとめることも可能) */
```

【課題12-3】

[mainでの処理]

```
printf("5->J: ");  
show_chars('5', 'J');  
printf("d->T: ");  
show_chars('d', 'T');
```

[実行結果]

```
5->J: 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J  
d->T: d c b a ` _ ^ ] \ [ Z Y X W V U T
```

【課題12-4】

グローバル宣言されたchar型の配列strに対して、char型の引数cと同じ文字がstrに含まれている個数を返す関数count_char()を作成してください。

(配列strに文字列を格納する際は、宣言時に初期値を代入しても良いし、scanf()で入力しても良い。)

[この関数のプロトタイプ宣言]

```
int count_char(char c);
```

```
/* 数えた個数を格納するための変数を宣言しておく */
```

```
/* 配列strに対する繰り返し処理をつくり、その中で、  
   それぞれの文字とcが等しいかどうかを比較して、  
   等しい場合は、数えている個数を増やす */
```


【課題12-4】

[配列strをグローバル変数として宣言する]

```
char str[100] = "Goooooooooooooogle";
```

[mainでの処理（配列strに"Goooooooooooooogle"が格納されている場合）]

```
printf("count_char: %d\n", count_char('o'));
```

[実行結果]

```
count_char: 10
```

まだ余裕のある人は…【課題12-5】

Point 2で示した、文字と文字コード（16進数）の対応表を出力する関数char_table()を作成してください。

[この関数のプロトタイプ宣言]

```
void char_table();
```

```
/* 0x20から1を加算しながら文字の出力を繰り返す。 */
```

```
/* 16文字おきに改行を出力する。 */
```

```
/* 可能ならば、どの16進数に対応するのかわかるように、
```

```
   行ヘッダ/列ヘッダも出力してみる（Point 2の出力参照） */
```

【課題12-5】

[mainでの処理]

```
char_table();
```

[実行結果]

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-a	-b	-c	-d	-e	-f
2-	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

小テストについて

小テストの注意点

- 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）
- 小テスト中は、演習室外へのネットワークアクセスは遮断される。

小テストについて

小テスト中に参照できるもの

- 教科書, 配付資料
- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
- 小テストでは紙媒体のものは参照可能
- 上記以外の情報を参照することは不正行為とする
例：USBで接続された機器に保存されているファイルの参照
ネットワークを介した情報の参照、など