COMP1511 19T1
Week 6, Tuesday: Meaning and Representation

Jashank Jeremy
`jashank.jeremy@unsw.edu.au`

characters, strings, text
references and indirection

Administrivia

Don’t panic!

Assignment 1: Coco
out now … start soon, or forever receive the Douglas!
extra help sessions now on (Mon AM, Thu AM, Fri PM),
see WebCMS 3 for details

Weekly Test #3
due *tomorrow*, 27 March 23:59:59

No Marc!
on week06tue, week06thu, week07tue
lectures by Jashank, instead.

Numbers

77 97 114 99 32 82 111 99 107 115

Numbers

77 97 114 99 32 82 111 99 107 115

these integers have no meaning implied by this representation

Numbers in Context

ASCII And You Will Receive

Our computers and programs help us add context and meaning.
For example, this ASCII table —

00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0a	NL	0b	VT	0c	NP	0d	CR	0e	SO	0f	SI
10	DLE	11	DC1	12	DC2	13	DC3	14	DC4	15	NAK	16	SYN	17	ETB
18	CAN	19	EM	1a	SUB	1b	ESC	1c	FS	1d	GS	1e	RS	1f	US
20	SP	21	!	22	"	23	#	24	$	25	%	26	&	27	’
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	‘	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c	|	7d	}	7e	~	7f	DEL

gives us one interpretation of values

Numbers in Context

ASCII Silly Question, Get A Silly Answer

**You Do Not Need To
Memorise The ASCII Table.**

There's absolutely no point in doing so;
manual entry *ascii(7)* ... or build your own.

You should use character literals where possible:

$$\texttt{'A'} = 41_{16} = 65_{10} = 101_{8} = 0100\,0001_{2}$$

(RULE Avoid magic numbers.)

Numbers in Context

All we store is bits.

Context and interpretation add meaning.

New Stuff

- `'.'` — single-quotes gives a character literal
- `"…"` — double-quotes gives a string literal

- `printf("%c", ch);`
  format code `"%c"` lets us print a single character

- `void putchar(int ch);`
  `putchar` outputs the single character *ch* to *standard output*
- `int getchar(void);`
  read one character from *standard input*; return it
  returns `EOF` if end-of-input was reached

Refactoring ‘print_char_array’

How Long Is A Piece Of String? (I)

```
void print_char_array (char array[]) {
    int i = 0;
    while (???) {
        putchar (array[i]);
        i++;
    }
}
```

How do we know when to stop?
How do we know how long the array is?

Refactoring ‘`print_char_array`’

How Long Is A Piece Of String? (II)

We need to know when to stop.
We can do this by knowing `(start, length)`,
or we can have a *sentinel* value to mark the end.

By convention,
we use the NUL character, `'\0'`,
to denote the end of a string.

String functions usually only take
the start of an array of chars,
and assume there will be
a NUL character at the end.

‘string_length’

How Long Is A String, Anyway?

```c
// Calculates the length of the string in `array`,
// excluding the terminating NUL byte ('\0').
int string_length (char array[]);
```

Arrays as Parameters

Passing Arrays Into Functions

```
char str[] = "Marc Rocks!";

string_length(str);
```

We don’t need square brackets.
We don’t need to index into the array.

Arrays as Parameters

Passing Arrays Into Functions

We’ve passed a *reference*, not the array itself.
This reference allows *mutable* access:
we can change the values in the array.