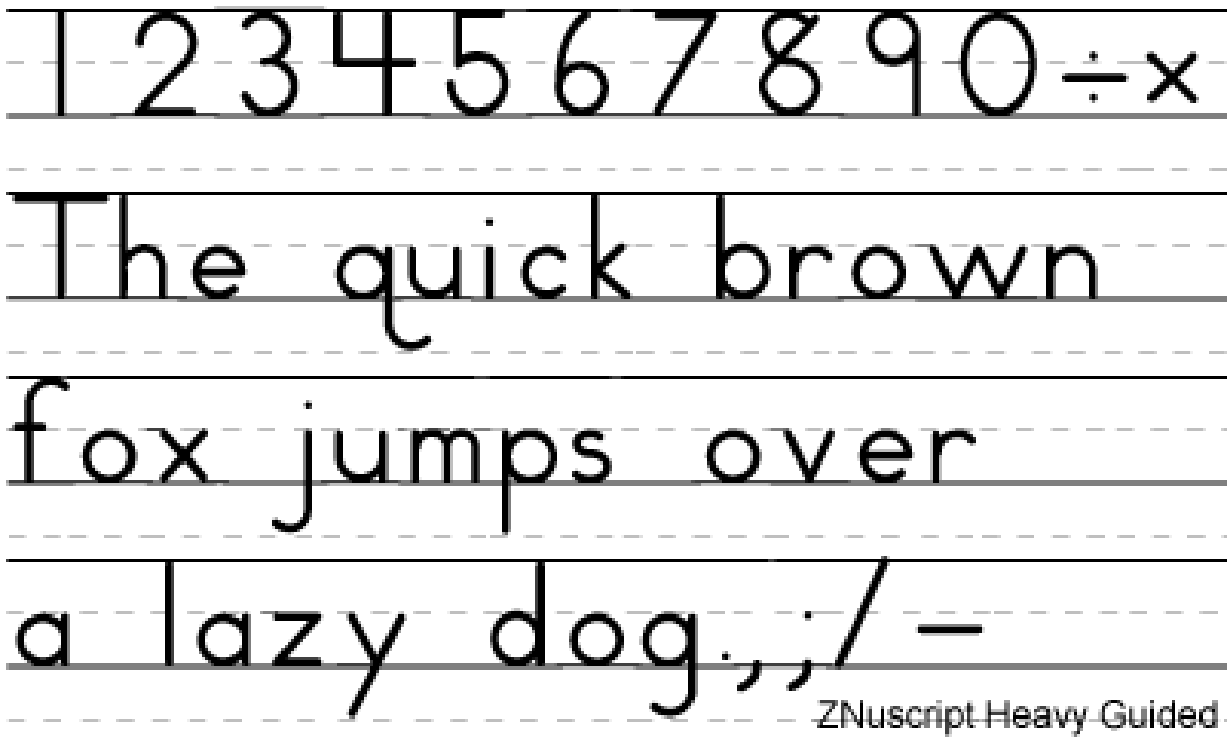
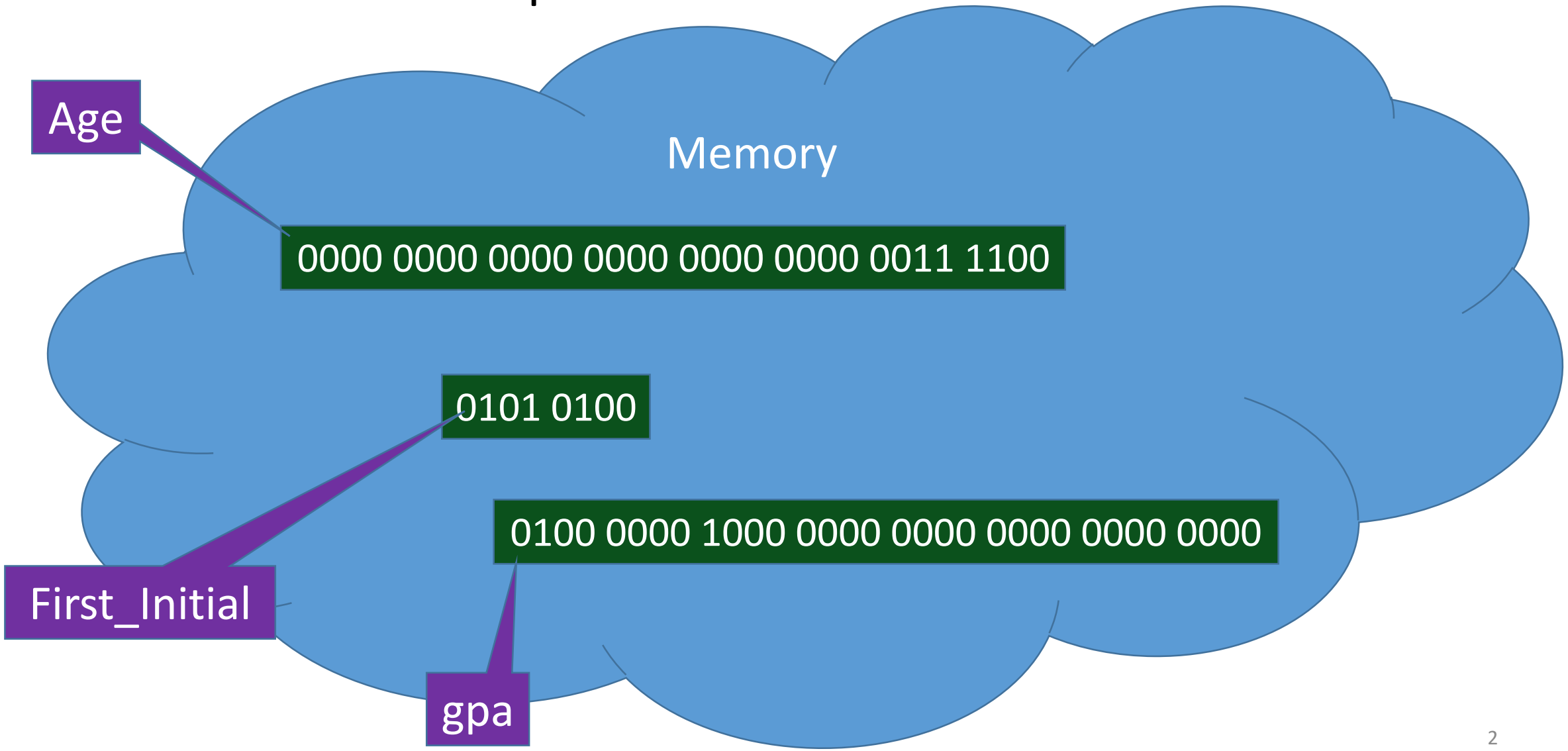
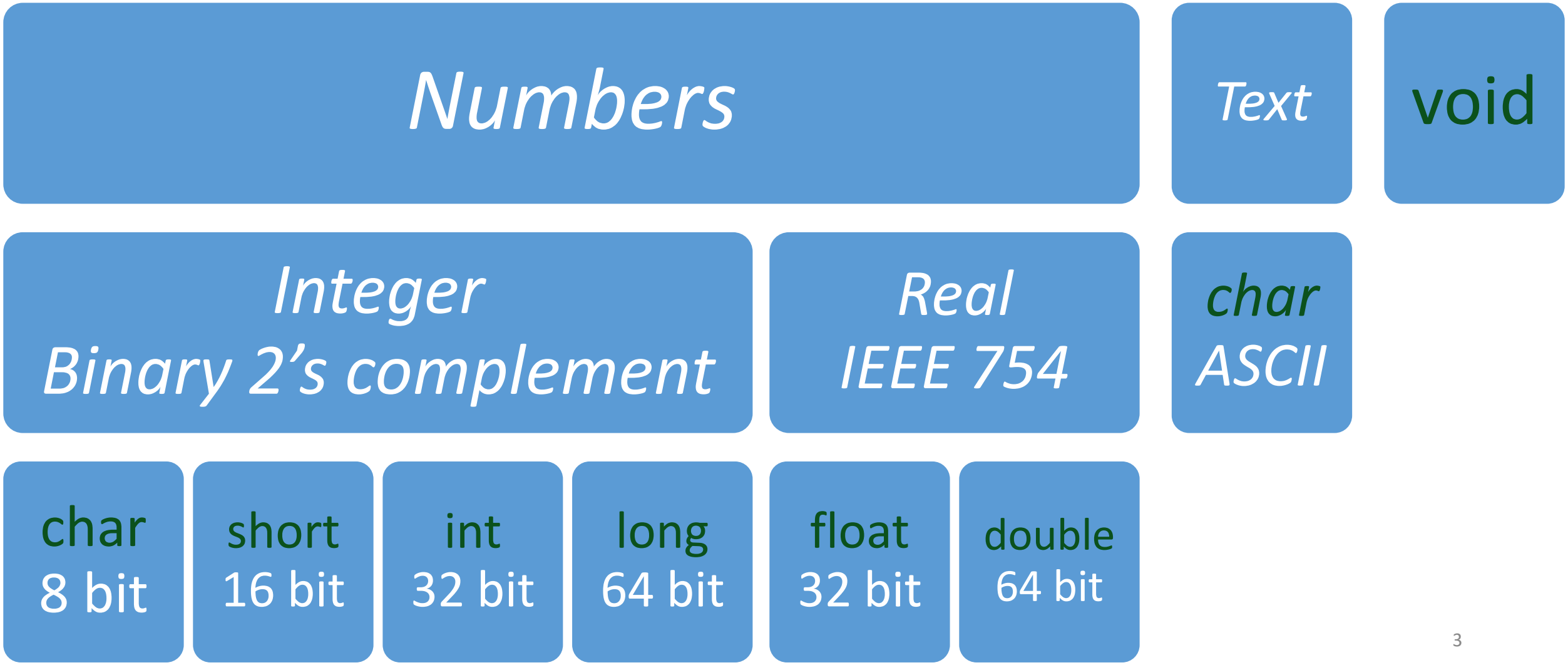


Data in C: Characters

Variable Concept



C Built-in Types



ASCII Character Coding

Chap 2.1.4

- ASCII – American Standard Code for Information Interchange
- Mapping from numbers 0-127, to characters

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Constants

- Single quotes surround a single character: `char fi = 'T';`
- Double quotes surround an array of characters: `char nm[4]= "Tom" ;`

- By the way… I forgot to talk about float constants
  - Any constant with a decimal point is assumed to be `float` : `float x=0.0;`
  - Use 'e' to indicate "times 10 to the power" : `float ac = 6.022e23;`
  - Use the "D" suffix to interpret as `double` : `double d = 1.7e200D;`

What happens when types are “mixed”?

- Mixed Type Expressions

```
char x='a'; int y; y=x+1; printf("%c\n",(char)y); // prints 'b'
```

- In an arithmetic context, use the ASCII code associated with the character!
  - ‘a’ is 97. y is 98, which is ASCII for ‘b’

C idiom: (using leaks) Alpha index

- Since alphabet is contiguous in ASCII, use arithmetic on ASCII codes

```
int alfaIndex(char x) {
        return 'a'-tolower(x);
}
char alfaAt(int y) {
        if (y<0 || y>25) return ' ';
        return 'a'+y;
}
```

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

C Idiom: (using leaks) Upper/Lower Case

- In ASCII, there is a 1 bit difference between the hex representation of an uppercase letter and a lower case letter… (0x20 on in lower case, off in upper case – 0x20 is a blank)

```
char toUpper(char x) {
        return x&0xDF;
}
char toLower(char x) {
        return x|' ';
}
```

*	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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	{	|	}	~	□

Notes on ASCII

- Normal file IO is in ASCII (or UNICODE which is a superset of ASCII)
  - Files not intended to be read or written by humans may be “binary” files
- Most editors can ONLY edit ASCII (or UNICODE) files!!!!
- Printers print using ASCII
  - Hence, ASCII codes which do NOT translate to a symbol are “unprintable” characters
  - Some unprintable characters translate to printer “commands”… e.g. \n for newline, \t for tab, \b for bell
- Keyboards translate to ASCII
- Terminals translate ASCII to symbols

Resources

- Wikipedia ASCII: https://en.wikipedia.org/wiki/ASCII
- Unicode ASCII subset: https://www.unicode.org/charts/PDF/U0000.pdf