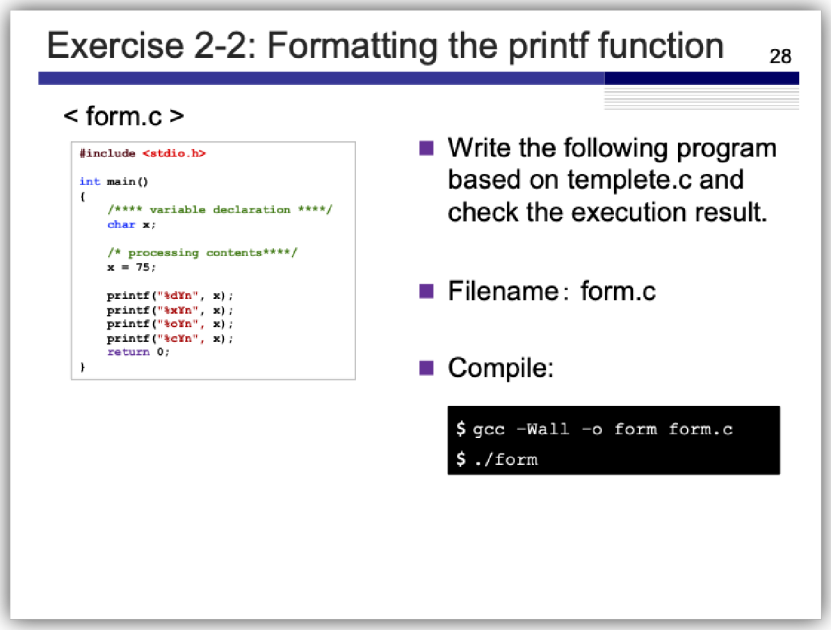
Exercise 2-2: form.c

Maximilian Fernaldy - C2TB1702



form.c highlights an important concept in C, where a `char` type can be converted into other types by using the `printf()` function. This program defines the `char x` as `75` and prints out 4 lines of output that at first glance seems random and unrelated to the original value. However, there is logic behind these values and there are practical uses for this technique later on.

Compiling and running the program yields the following output in the terminal:

The first line prints out `x` formatted as an integer. As expected, the program prints out 75, but this is different from the 75 that's stored in `x`. Even though we did assign the value `75` to `x`, this is not exactly what is stored in the memory, as we will see later. Here, `printf()` formats what is stored in `x` to an integer and then prints out the formatted value to the terminal as output.

The second line is more interesting. Unexpectedly, it prints out `4b`, which doesn't seem to make sense at first. However, if we know what type it was formatted as, we can begin to understand it. Taking a look at `form.c`, we know that the second line

prints out `x` after formatting it as a hexadecimal (base-16) number, instead of the usual decimal (base-10). This means we have an extra 6 set of values before moving on to the next digit, as such:

Hexadecimal	0	1	2	...	9	a	b	c	d	e	f
Decimal	0	1	2	...	9	10	11	12	13	14	15

To get numbers larger than 16, we just insert a new digit. For example, 28 is 1c in hexadecimal, because c carries the value 12 and $1 \times 16 + 12 = 28$. Now we can intuitively understand why 75 was printed as 4b, because b carries the value 12 and $4 \times 16 + 11 = 75$. Each digit's value is multiplied by a power of 16, and this carries over to the alphabetical digits too. For example, the hex of 228 is e4, because e carries the value of 14 and $14 \times 16 + 4 = 228$.

We can imagine this being very useful when trying to make a program that involves hexadecimal color codes, because it means we can convert between decimal values (from 0 to 255) and hexadecimal values (from 0 to ff).

The third line is similar in that it's using a different numerical system than base-10, but it's in octal. As the name suggests, it is base-8, which means it converts `75` stored in `x` into 113, because $1 \times 8^2 + 1 \times 8^1 + 3 \times 8^0 = 75$. Again, each digit is multiplied by 8 and added to the sum to represent the final value.

The fourth and final line is also quite interesting, because it reveals what happened when we created the variable `x` and assigned `75` to it. By using the format specifier `%c`, we format the value stored in `x` as a single character, and then print it out. `75` is then converted to `K`, because it is the equivalent value for type `char` in the ASCII (American Standard Code for Information Interchange) standard. Other equivalent values can be seen below:

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]