Supplementary material (CS101)

Data Types

Memory for variables is allocated based upon their 'type'

int number (2 bytes)

_ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _

Let us say I want to store the value '1' in 'number'.
My domain is (1-10).
The equivalent binary representation for '1' will be:
00000000 00000001

Inefficient usage of memory

- int

- short int

- unsigned short int

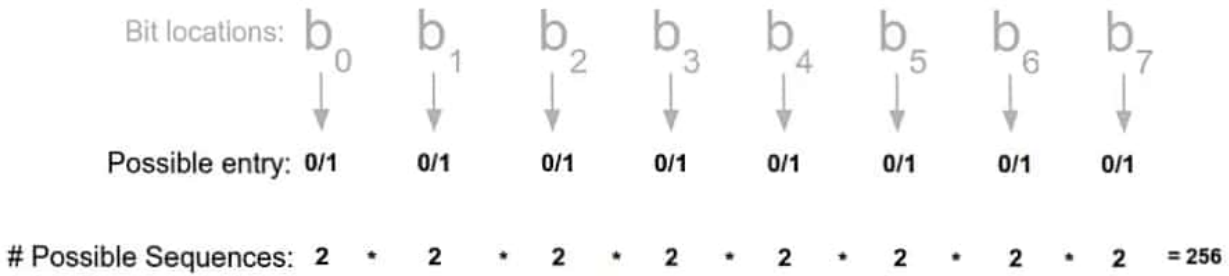
Data types (ANSI C)

Data Type	Size (bytes)	Range of Values	Format Specifier
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short int	1	-128 to 127	%hi
unsigned short int	1	0 to 255	%hu
int	2	-32,768 to 32,767	%d
unsigned int	2	0 to 65,535	%u
long int	4	-2,147,483,648 to 2,147,483,647	%l
unsigned long int	4	0 to 4294967295	%lu
float	4	3.4E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	10	3.4E-4932 to 1.1E+4932	%Lf

Please note, above mapping is for 16 -bit system. Sizes and ranges vary across systems (16 bit/ 32 bit/ 64 bit). Also, using '%f' for printing int data may lead to errors in certain configurations.

Range (Unsigned short int)

Size = 1 byte = 8 bits



Range of unsigned short int is 0 to 255

Range (Short int)

Signed: Also includes -ve numbers

- 0 MSB -> +ve, 1 MSB -> -ve
- Storage requires binary representations
- Binary representations for positive number is calculated using decimal to binary conversion
- -ve numbers are converted to binary representations using 2's complement (1's complement + 1) of the +ve counterpart
- 1's complement simply inverts each bit (1 to 0, 0 to 1)
- Example:
  - 1 -> 00000001
  - -1 -> 2's complement of 1 -> (1's complement of 00000001) + 1 ->11111110 + 1 -> 11111111
- Max +ve: 01111111 -> 127
- Min -ve: 10000000 -> 01111111 + 1 -> 10000000 -> -128

Range of short int is -128 to 127

ASCII Table

- American Standard Code for Information Interchange
- ASCII codes are used to represent characters/symbols/actions uniformly across different systems
- ASCII codes map characters/symbols/actions to numbers.
- Each character/symbol/action will have an equivalent number (code) that can be used for storing the data.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

```
#include <stdio.h>

int main()
{
char ch1 = 'B';
char ch2 = 66;
printf("%c\n",ch1);
printf("%c",ch2);
return 0;
}
```

Output:

B

B

strcpy()

- A function defined in the 'string.h' header file
- Copies the content of one string to the other
- First parameter is the destination, second parameter is the source

```
#include <stdio.h>
#include <string.h>

void main() {

  char superhero[15] = "Bruce Banner";
  char angry[15] = "Hulk!";
  printf("%s\n",superhero); //print the default content of string 'superhero'
  strcpy(superhero, angry);//copy the content of string 'angry' to sting 'superhero'
  printf("%s",superhero); //print the new content of string 'superhero''

}
```

Output:

Bruce Banner
Hulk!

Taking input string including spaces

Using scansets:

- scanf process data according to scanset (%[])
- %[^\n] allows the read operation till the first 'new line' occurrence
- %*c reads the new line and discards it

```
#include <stdio.h>

void main()
{
   char answer[50];
   printf("What is your name?\n");
   scanf("%[^\n]%*c", answer); //read input until a new line, discard new line
   printf("%s", answer);
}
```

Terminal:
What is your name?
My name is Suraj. ← Input
My name is Suraj. ← Output

Format specifier (octal and hexadecimal)

- Octal and hexadecimal representations of data can be handled using **%o, %x, %X**.

$1*8^1 + 3*8^0 = 11$

```
#include <stdio.h>

void main()
{
    int val;
    scanf("%o",&val);
    printf("Decimal representation: %d\n",val);
    printf("Octal representation: %o\n",val);
    printf("Hexadecimal representation: %x\n",val);
    printf("Capitalised Hexadecimal representation: %X\n",val);


}
```

Input:
13

Output:
Decimal representation: 11
Octal representation: 13
Hexadecimal representation: b
Capitalised Hexadecimal representation: B

%p vs %d format specifier

%p is used to print the address

```
#include <stdio.h>

void main()
{
    unsigned long int val = 11;
    printf("Value: %lu\n",val);
    printf("Address: %p\n",&val);
    printf("Address (decimal equivalent): %lu",&val);


}
```

Output:

main.c:10:45: warning: format '%lu' expects argument of type 'long unsigned int', but argument 2 has type 'long unsigned int *' [-Wformat=]

Value: 11
Address: 0x7fff94268658
Address (decimal equivalent): 140735678940760

Warning