EECS 2031

I/O

Topics

- Functions
- User Interface

Data Types

- 4 basic types in C
  - char – Characters
  - int -- Integers
  - float – Single precision floating point numbers
  - double – Double precision floating point numbers
  - Unsigned, long, long long, short (different from system to system).

In <stdint.h>

- int8_t unit8_t
- int16_t unit16_t
- int32_t unit32_t
- int64_t unit64_t

C Basics

- Variable name is a combination of letters, numbers, and _ that does not start with a number and is not a keyword
- Abc abc5 aA3_ but not 5sda
- #include <filename.h> replaces the include by the actual file before compilation starts
- #define abc xyz replaces every occurrence of abc by xyz

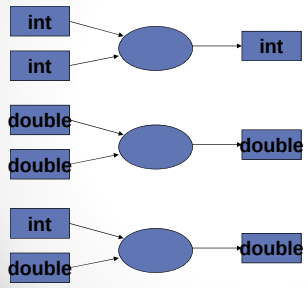
Basic Operations

- Addition, subtraction, multiplication, division, and mode (+ - * / %).

C Basics

- Decimal numbers 123487
- Octal: starts with 0 0654
- Hexadecimal starts with 0x or 0X ox4Ab2
- 7L for long int =7
- 8U for unsigned
- For floats 24, 23.45, 123.45e-8, 3.4F, 2.15L

Mixed type arithmetic




```
int x=5, y=2, w;
double z, q = 2;

z = x/y;
        // z = 2.0
w = x/y;
        // w = 2
z = x/q;
        // z = 2.5
w = x/q;
        // w = 2
```

Mixed type arithmetic

- 17 / 5
  - 3
- 17.0 / 5
  - 3.4
- 9 / 2 / 3.0 / 4
  - 9 / 2 = 4
  - 4 / 3.0 = 1.333
  - 1.333 / 4 = 0.333

Mixed type arithmetic

- How do you cast variables?

e.g.

```
int varA = 9, varB = 2;
double  varC;

varC = varA  /  varB;      //  varC is 4.0

varC = varA  /  (double) varB   // varC is 4.5
```

Doesn't change the value of varB, just changes the type to double

C Basics

- Expressions
- abc= x+y*z
- J=a%i
- ++x vs. x++
- X += 5;
  // x = x + 5;
- Y /= z;
  // Y = Y / z

What is x *= y+1 ? addition higher than *=

Pre- and Post- Operators

- ++ or --
- Place in front, incrementing or decrementing occurs BEFORE value assigned

i = 2 and k = 1

k = ++i;

```
i = i + 1; 3
k = i;     3
```

k =--i;

```
i = i - 1; 1
k = i;     1
```

- Place in back, occurs AFTER value assigned

i = 2 and k = 1

k = i++;

```
k = i;     2
i = i + 1; 3
```

k = i--;

```
k = i;     2
i = i - 1; 1
```

Precedence

( )	Parentheses	L to R	1
++, - -	Postincrement	L to R	2
++, - -	Preincrement	R to L	3
+, -	Positive, negative	L to R	3
*, /, %	Multiplication, division	L to R	4
+, -	Addition, subtraction	L to R	5
<=, >=, >, <	Relational operator	L to R	6
==, !=	Relational operator	L to R	7
&&	Logical AND	L to R	8
||	Logical OR	L to R	9
+=, -+, *=, /=, %=	Compound assignment	R to L	10
=	Assignment	R to L	10

Examples

- int a=2, b=3; c=5, d=7, e=11, f=3;
- f +=a/b/c; 2/3=0/5=0 --> f=3
- d -=7+c*--d/e; 5 * 6 = 30; 30/11=2 2+7=9; d=d-9= 6-9=3
- d= 2*a%b+c+1; 2*2=4%3=1; 1+5+1=7
- a +=b +=c +=1+2; c=3+5=8; b=3+8=11; a=2+11=13

C – First Program

```
1. /* Our first program  */
2. #include <stdio.h>
3.  void main() {
4.              printf("Hello World \n");
5.  }
```

Special ,Characters

\n	New line
\t	Tab
\"	Double quote
\\	The \ character
\0	The null character
\'	Single quote

Formatting Output

```
printf("|%d|%5d|%-5d|%5.3d\n",i,i,i,i);
```

```
|40|   40|40   |  040
```

```
printf("|%10.3f|%-10.3f|%f|%g|%e\n",x,x,x,x,x);
```

```
|     8.100|8.100     |8.100000|8.1|8.100000e+00
```

Modifiers

- signed (unsigned) int long int
- long long int
- int may be omitted
- sizeof()

Input

- Scanf is used to read from the standard input
- `scanf("%d  %d\n",&i,&j);`
- `scanf("%d%d\n"),&i,&j);`
- `scanf("%d,%d\n"),&i,&j);`
- `scanf("%d, %d\n"),&i,&j);`

Characters

- One byte
- Included between 2 single quotes
- char x ='A'
- Character string "This is a string"
- 'A' != "A"

- X='\012' newline or 10 decimal

A	A	\0

Characters

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

Boolean Expressions

- Relational operators
- ==, !=, <, <=, >, >=
- Logical operators
- &&, | |, !

I/O

- Every program has a standard input and output (stdin, stdout and stderr)
- Usually, keyboard and monitor
- Can use > and < for redirection
- printf("This is a test %d \n",x)
- scanf("%x%d",&x,&y)

%d %s %c %f %lf

integer string character float double precision

I/O

- `int getchar`
  - Returns the next character on standard input or EOF if there are no characters left.
- `int putchar(int c);`
  - Writes the character c on the standard output
- `int printf(char *format,…)`
- `printf("The result is %f \n",x);`

Bitwise Operators

- Works on the individual bits
- &, |, ^, ~
- short int i=5, j=8;
- k=i&j;
- k=i|j;
- k=~j;

Bit Shifting

- x<<y means shift x to the left y times
- x>>y means shift x to the right y bits
- Shifting 3 many times

0	3
1	6
2	12
3	24
4	48
13	49512
14	32768

Bit Shifting

- What about left shifting
- If unsigned, 0 if signed undefined in C
- It could be logical (0) or arithmetic (sign)
- Unsigned int I =714
- 357 178 89 44 22 11 5 2 1 0
- What if -714
- -357 -178 -89 . . . -3 -2 -1 -1 -1 -1