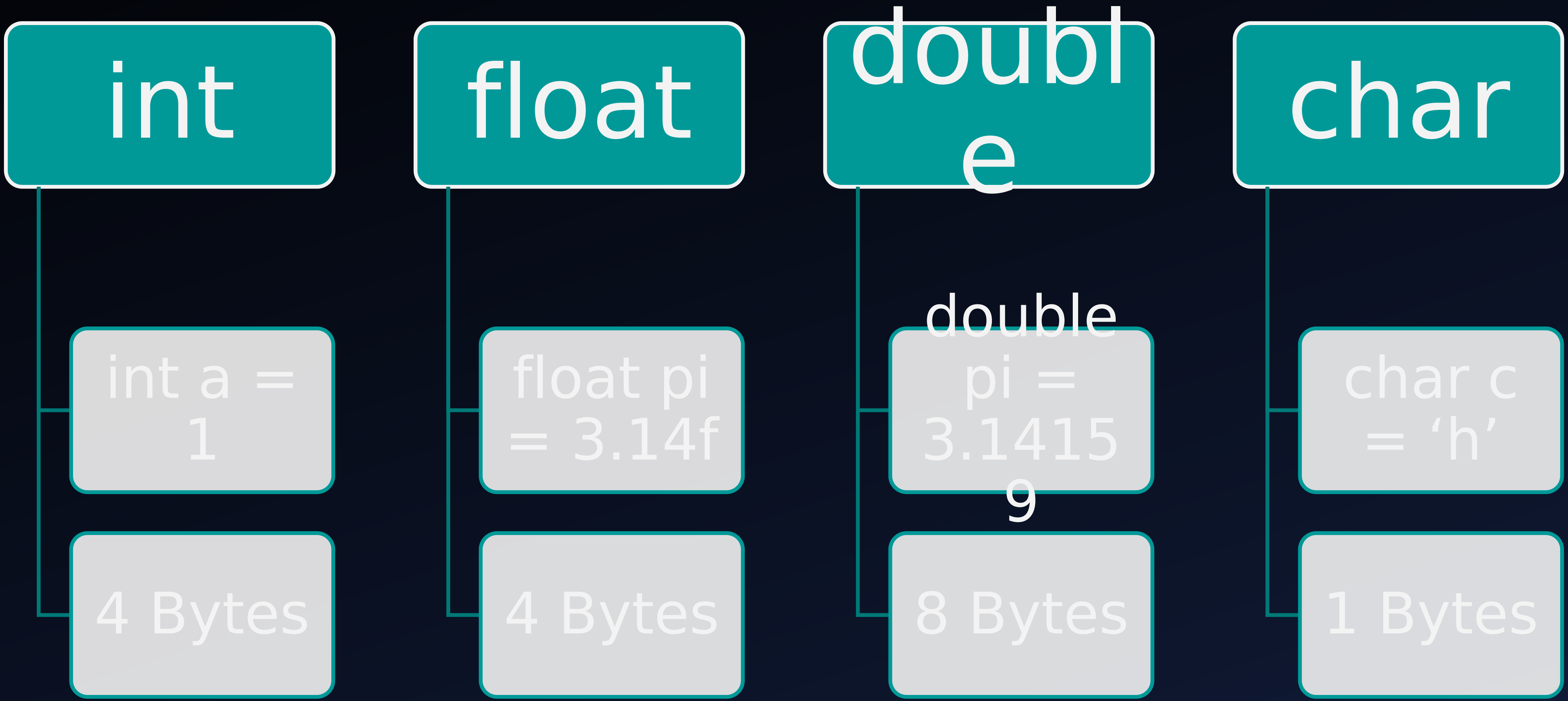
C Programming Language

December 5, 2014

Today’s task

- Basic C Features
  - Basic types
  - Basic programming structure
    - Sequence
    - Branch
    - Loop
- Array
- C-style string

Basic types



int

int a = 1

4 Bytes

float

float pi = 3.14f

4 Bytes

double

double pi = 3.14159

8 Bytes

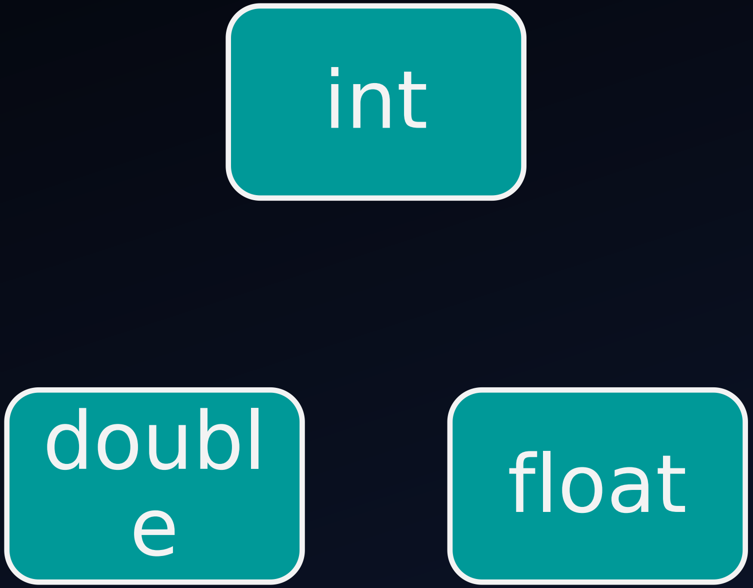
char

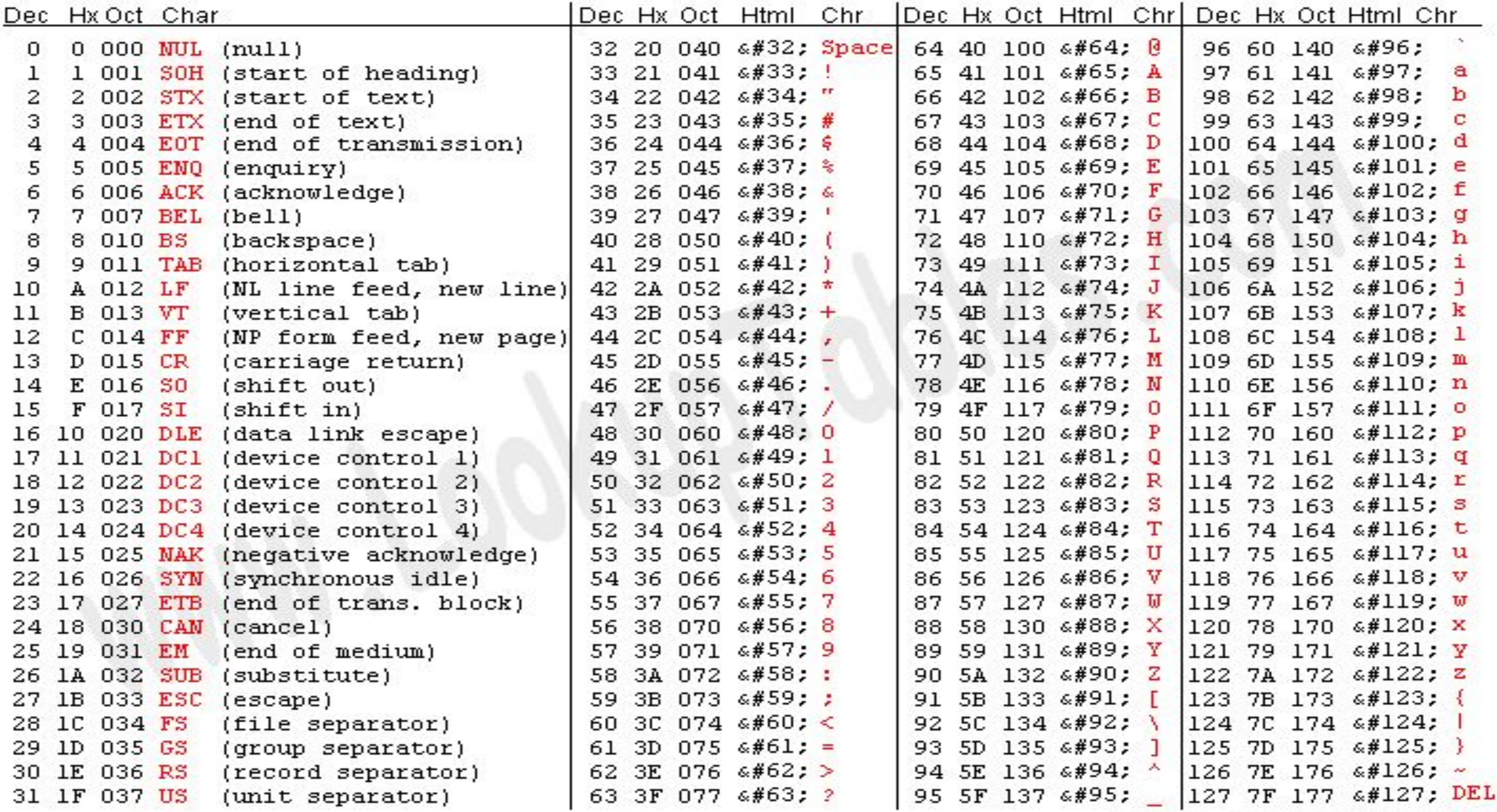
char c = ‘h’

1 Bytes

Type convert

- Accuracy
  - double > float > int
- int and char convert
  - Ascii table



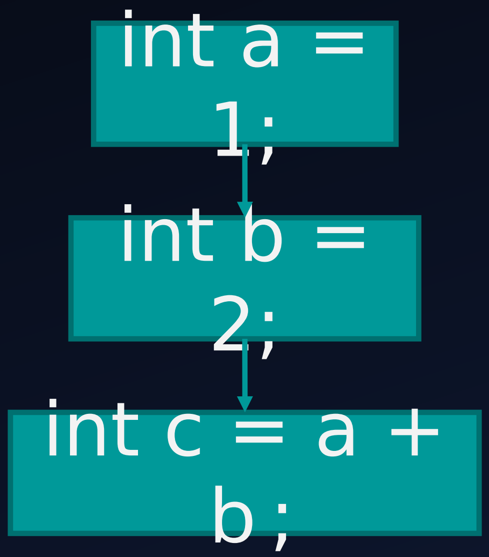


Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;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

Source: www.LookupTables.com

Three Basic program structure

- sequence
- branch
- loop



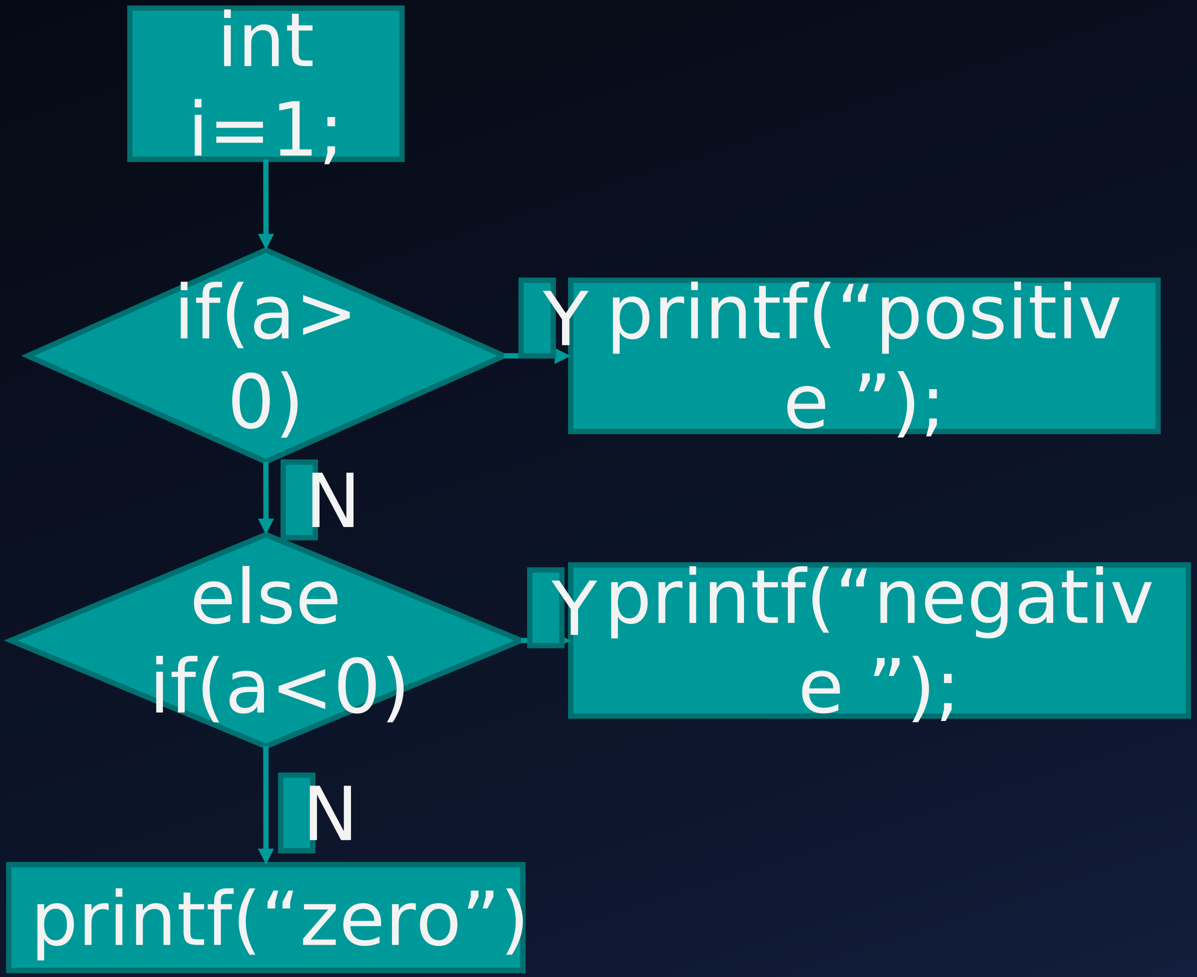
int a = 1;

int b = 2;

int c = a + b ;

Branch if···else

```
int a=1;
if(a>0)
{
	printf("positive");
}
else if(a < 0)
{
	printf("negative")
}
else
{
	printf("Zero")
}
```



int i=1;

if(a>0)

Y printf(“positive ”);

N

else if(a<0)

Y printf(“negative ”);

N

printf(“zero”)

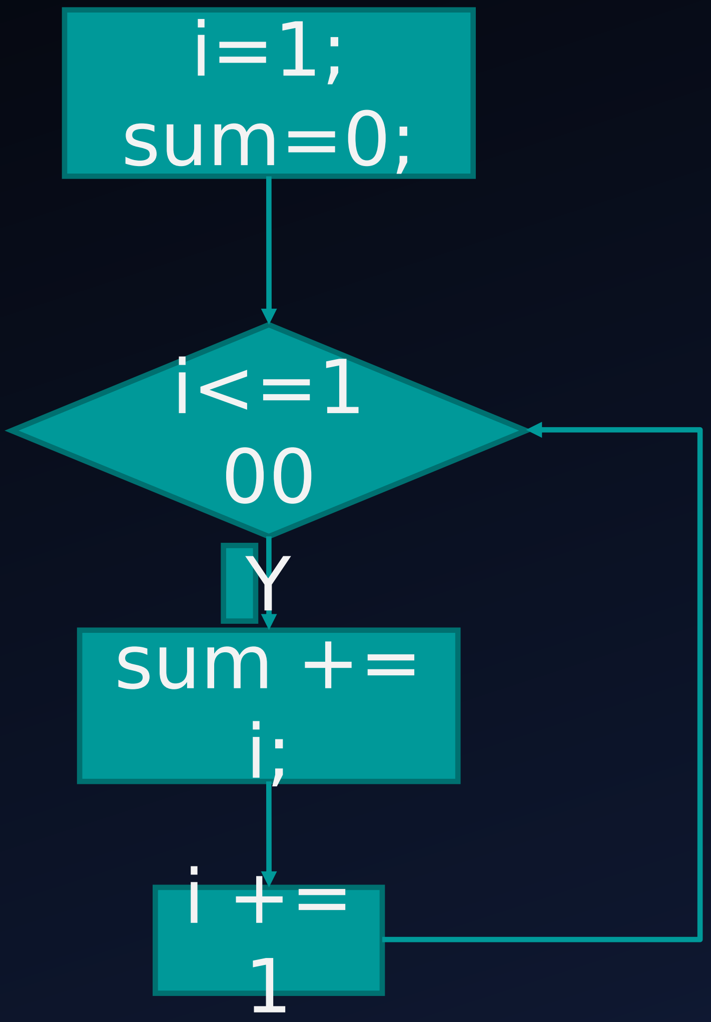
Loop

- Perform a set of repetitive task until text expression becomes false
  - while
  - do…while
  - for

- calculate 1+2+3+⋯+100

Loop while

```
int i = 0;
int sum = 0;
while(i<=100)
{
    sum += i;
    i += 1
}
```



i=1;
sum=0;

i<=1
00

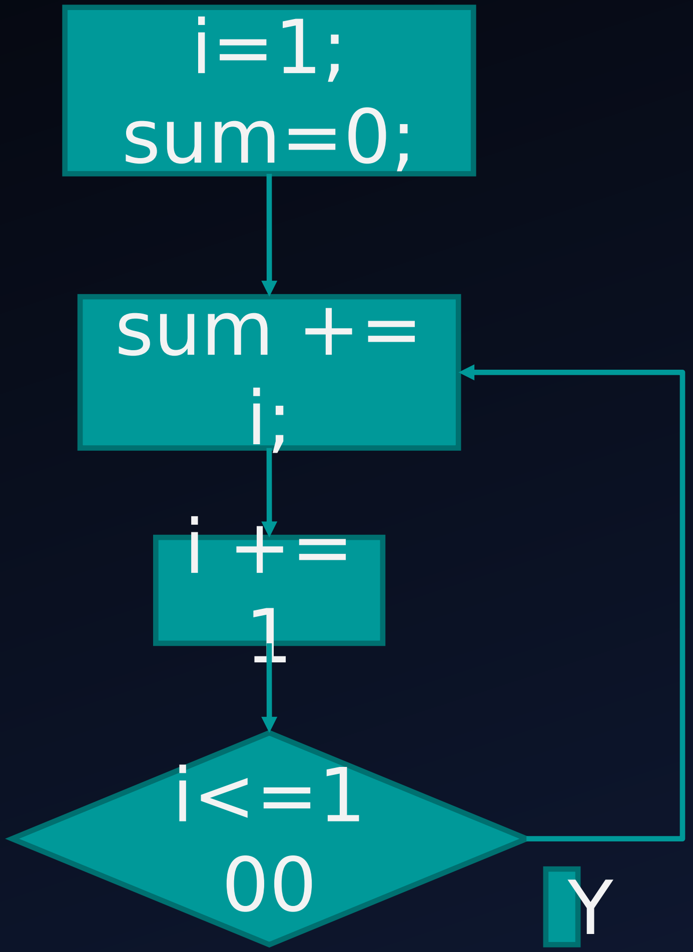
Y

sum +=
i;

i +=
1

Loop do while

```
int i = 0;
int sum = 0;
do
{
    sum += i;
    i += 1;
} while (i<=100);
```



Loop for

```
for (int i=0;i<=100;i++)
{
    sum += i;
}
```

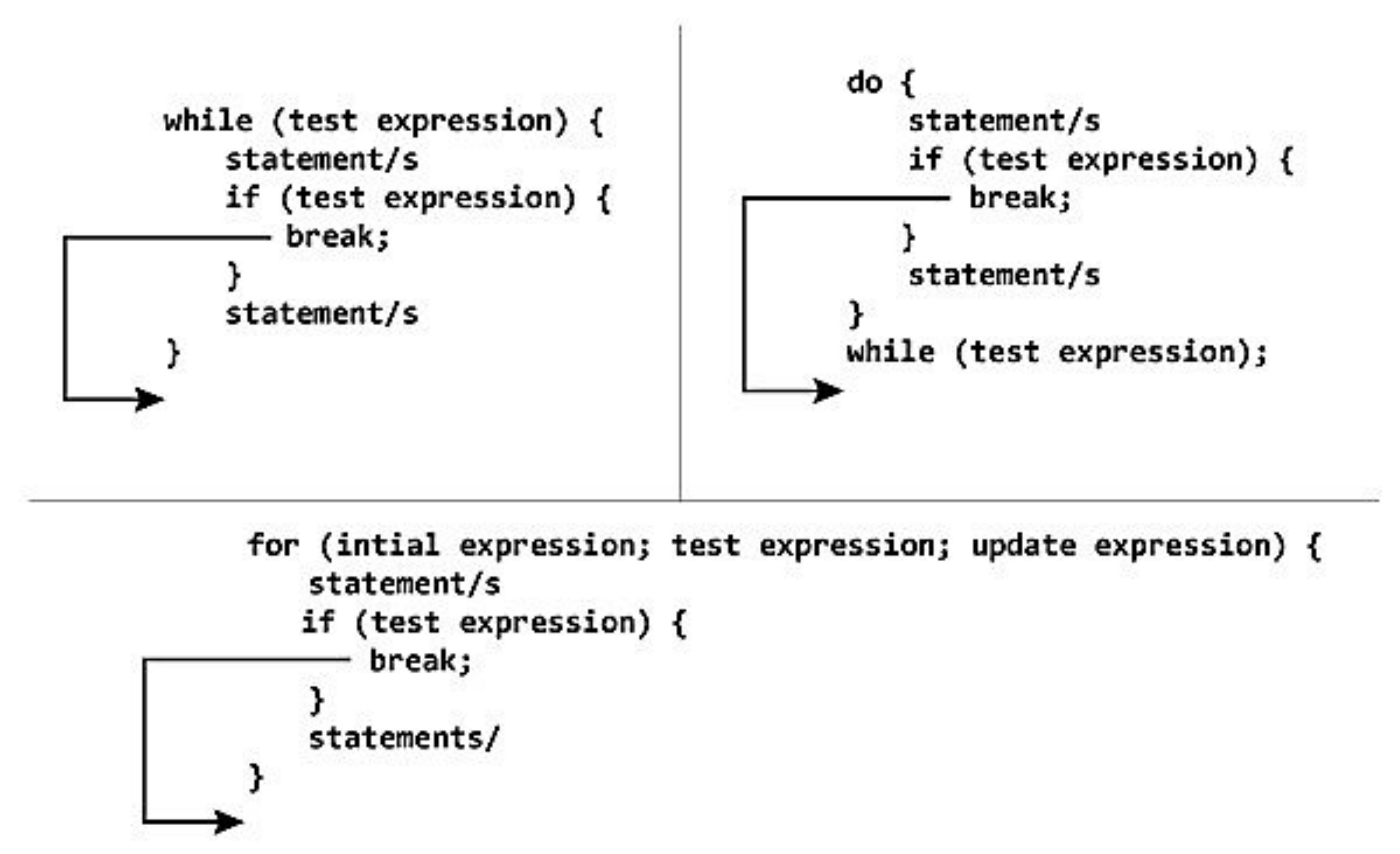
When should we use while, do while and for?

- most loops can be written in all three ways
- while
  - check the expression first and do jobs
- do⋯while:
  - do jobs first and then check the expression
- for
  - if you know the loop times exactly, use for
  - code is short

Break and continue

- Break: stop the loop

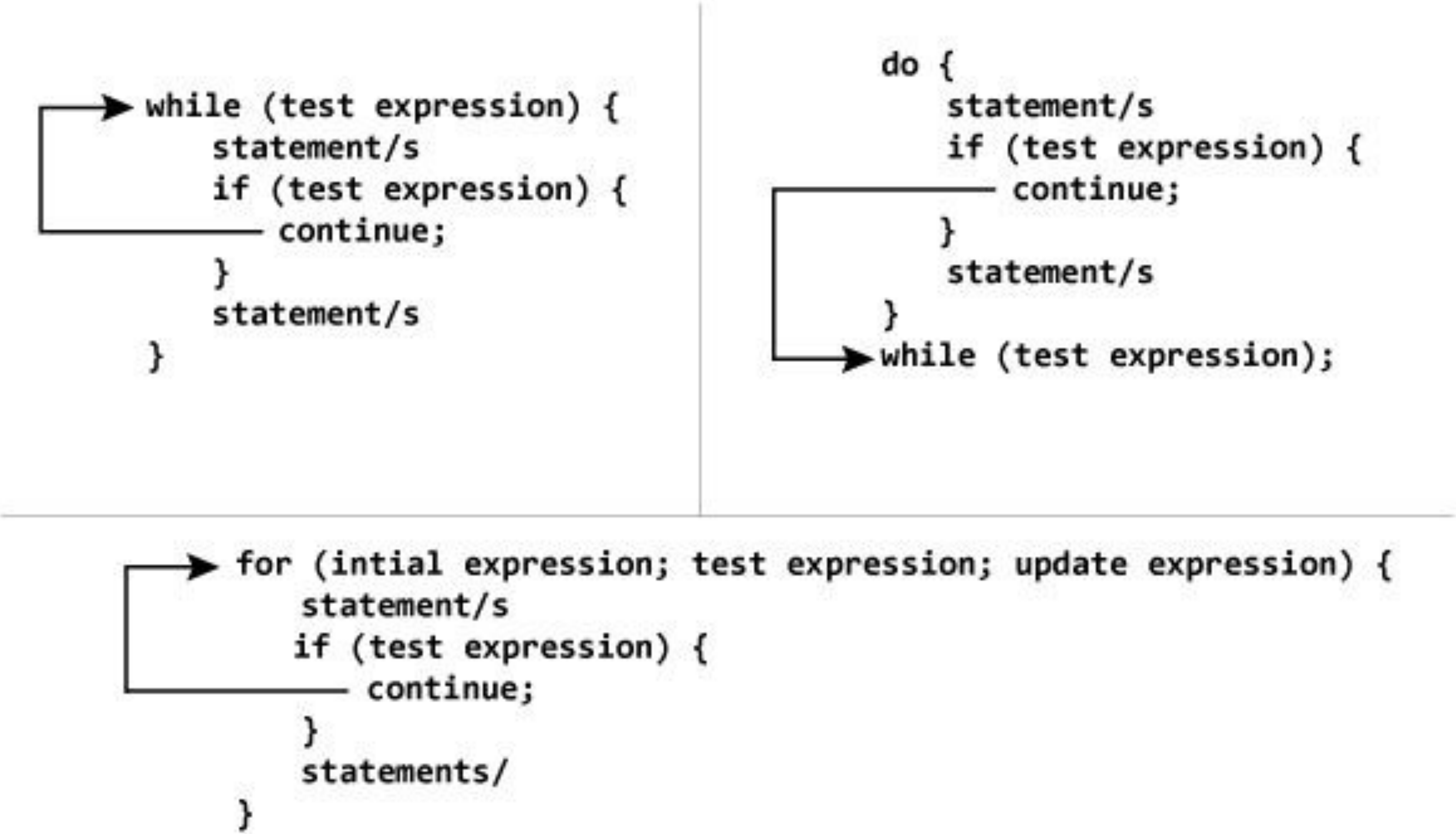
- Continue: only skip this time



```
while (test expression) {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        break;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        break;
    }
    statements/
}
```



```
while (test expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
```

```
do {
    statement/s
    if (test expression) {
        continue;
    }
    statement/s
}
while (test expression);
```

```
for (intial expression; test expression; update expression) {
    statement/s
    if (test expression) {
        continue;
    }
    statements/
}
```

function

- Code that we need to use more than once

```
int mul(int x, int y)
{
    return x * y;
}
```

Pass parameter by value or by address

```
int self_add1(int x)
{
    x = x + 1;
    return x;
}
```

```
int self_add2(int *x)
{
    *x = *x + 1;
    return *x;
}
```

```
printf("%d\t%d",self_add1(x),x);
printf("%d\t%d",self_add2(&x),x);
```

A little challenge

- Write a little program that users can type an expression and the program will return the result
- e.g.
  - if users type 1 + 2 and press enter
  - the program will display 3 on the screen

Array declare

```
int lst1[] =
{0,1,2,3,4,5,6,7,8,9};
int lst2[10];
int lst3[10] =
{0,1,2,3,4,5,6,7,8,9};


int twodim[10][10];
int threedim[10][10][10];
```

Array modify and use

- modify and access an element with index
  - index range: 0~n-1

```
int lst[10];
lst[3] = 4; // ok
lst[10] = 1; // wrong
```

Array and pointer

```
int
lst[10];
```

- lst stores the address of the first element of the array

- What’ s the difference between array and pointer?

- The address stored in the lst is unable to modify, but the store stored in a pointer is ok to modify

Disadvantage of array

- The size of array is set previously and unable to modify

C-style Strings

- what is a string?
  - “Hello, world!”
  - An array of char
- Declaration

```
char astring[50];
char *pstring[50];
pstring = (char*)
malloc(sizeof(char) * 50);
//...
free(pstring);
```

fgets function

```
char string[256];
printf( "Please enter a long
string: " );
/* notice stdin being passed in
*/
fgets ( string, 256, stdin );
printf( "You entered a very
long string, %s", string );
```

string manipulate function <string.h>

stsrcmp:	string compare
strcat:	string concatenate
strlen:	length of a string

Homework

- implement your own string manipulate function including
  - strlen
  - strcmp
  - strcat