Conditionals

Dr. Na Li
CSE @ UTA
Jan. 22, 2013

if statements

- if statement gives you choice of either executing a statement or skipping it
- The basic format of the if statement is

```
if (condition_is_true)
    do_something;
```

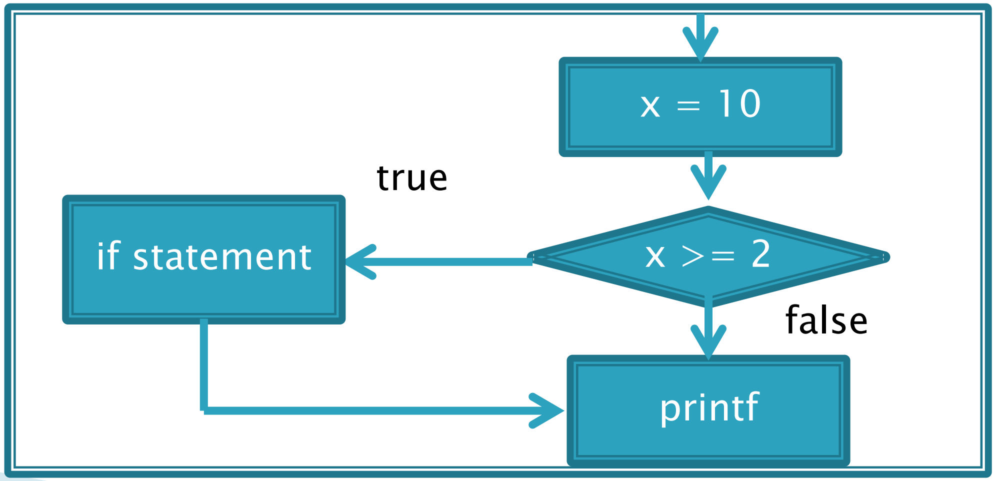
// *condition_is_true* should be an expression

//need the parenthesis for the expression

// If expression evaluates to true (nonzero), do something. Otherwise, it is skipped. Normally, expression is relational expression. But in general, any expression will work since it has a value, which can be mapped to true or false.

- Examples

```
int x;
int y = 6;
scanf (“%d”, &x);
if (x >= 2)
    y = 10;
printf(“y is %d.\n”, y);
```



Indent Style in C Programming

- indent style – a convention governing the indentation of blocks of code to convey the program's structure
- Visual Studio manages indent automatically by default
- Manually, “Tab” key on the keyboard will help

if statements cont.

- Handle more than one statements when the condition is true
- Create a block of statements by using braces.
  - Example

```
if (x >= 2)
{
      y = 10;
      printf("y is now %d", y);
}
next statement;
```

// with one statement, you can use "{}", but it's not necessary.

// without the braces what's going to happen here?

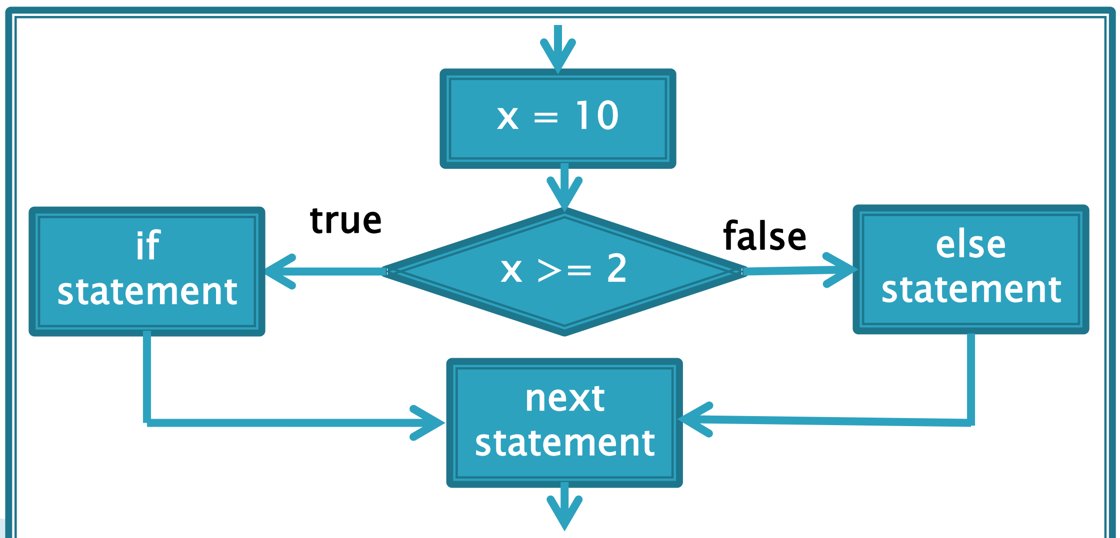
if and else statements

- To do one thing if a condition is true but another if the condition is false – if-else statement:
- Basic format of if-else statement

```
if (expression_is_true)
  do_something (if statement);
else
  do_something_else (else statement);
```

- Example:

```
int x = 10;
if(x>=2)
    printf("x>=2\n");
else
    printf("x<2\n");
your next statement;
```



if and else statements cont.

- else statement must have an if statement to match
- Not allowed to have any statement between if statement and else statement (except the cases of nested if-else)
- Example: (a wrong program!)

```
int x = 4;
int y;
if (x >= 2)
    y = 10;
    printf("y is now 10.\n");
else
    printf("y is not assigned.\n");
```

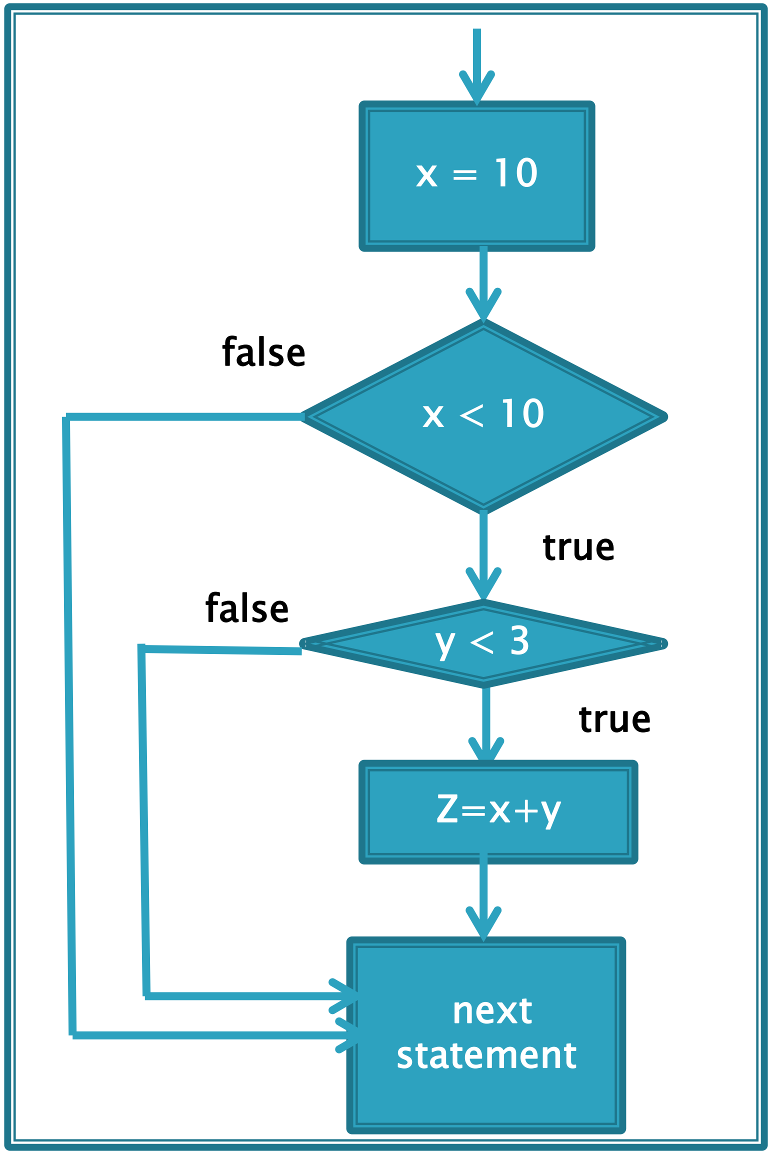
- else statement can also be a block of statements, but remember to give "{ }"
- Example:

```
else
{
    y = y + 2;
    printf("y is not
    assigned.\n");
}
```

Nested if

- Have a second conditional statement in when the first condition is true
- if statements can be nested
- Example :

```
    if ( x < 10 )
        if( y < 3 )
            z = x + y;
    your next statement
  comes here;
    // To assign a value of
  (x+y) to z only if x > 10 and
  y < 3.
```



Nested if and else

/* indentation is used to show correct logic; the ***else*** goes with the nearest unmatched ***if*** statement within a block , where block is defined by using braces.*/

- Examples: (nested in if statement)

```
(1) if (x == 13)
            if (y == 52)
                printf("Test1.\n");
            else
                printf("Test2.\n");
     printf("Test3.\n");
(2) if (x == 13)
          if (y== 52)
              printf("Test1.\n");
    else
          printf("Test2.\n");
    printf("Test3.\n");
 (3) if (x == 13)
     if (y== 52)
     printf("Test1.\n");
     else
     printf("Test2.\n");
     printf("Test3.\n");
```

Nested if and else cont.

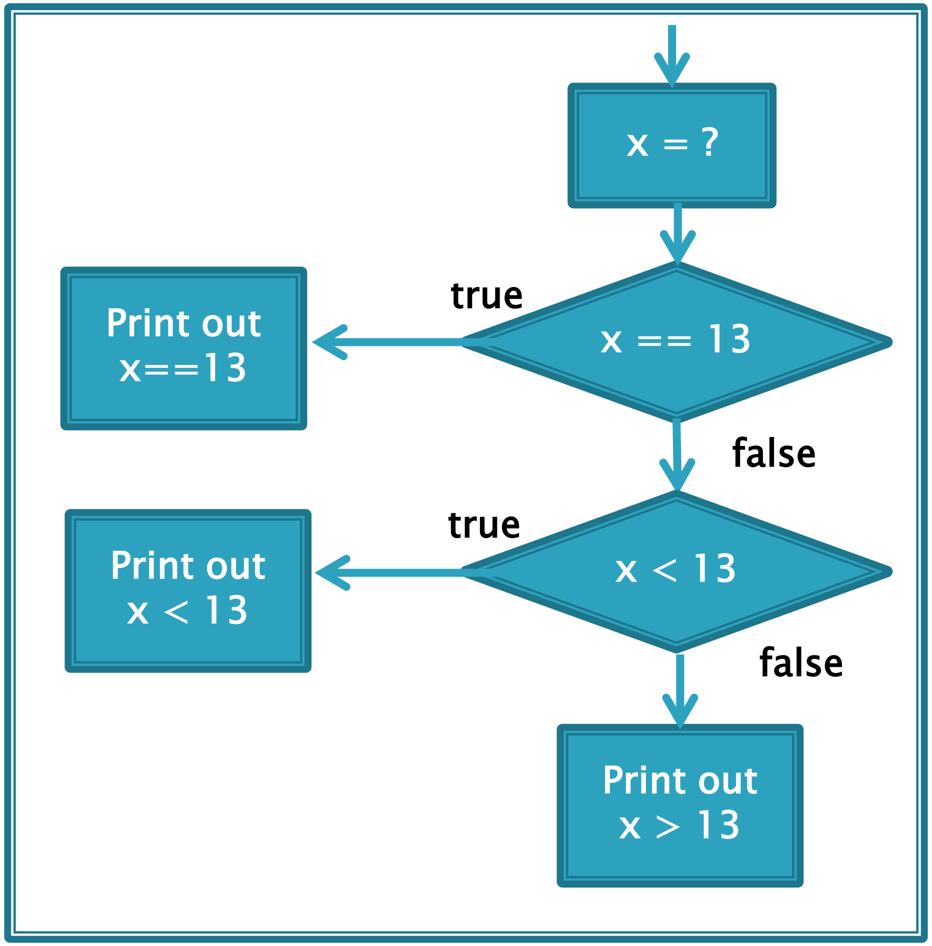
```
(4) if (x == 13)
    {
        if (y == 52)
            printf("Test1.\n");
        else
            printf("Test2.\n");
    }
    printf("Test3.\n");

(5) if (x == 13)
     {
        if (y== 52)
            printf("Test1.\n");
     }
     else
        printf("Test2.\n");
    printf("Test3.\n");
```

- (1)(2)(3)(4) are the same, but not (5)
- (2) and (3) are not good program styles, you should use indent to convey your code block, like what (1) does, or use the braces like the (4) example.

Nested if and else cont.

- Examples: (nested in else statement)
- (1) if (x == 13)
-     printf("x == 13.\n", x);
-   else if (x < 13)
-     printf("x < 13.\n");
-   else
-     printf("x > 13.\n");
- (2) if (x == 13)
-     printf("x == 13.\n", x);
-   else
-     if (x < 13)
-       printf("x < 13.\n");
-     else
-       printf("x > 13.\n");
- (1) and (2) are the same.



Nested if and else cont.

- Any expression that evaluates to a nonzero value is considered true.
- Examples:
  - if (-3.5)
    printf("non-zero values are true\n"); // this will be printed.
    else
    printf("this never prints\n");
  - if (0)
    printf("zero is false\n");
    else
    printf("this is always false\n"); // this will be printed.
- WARNING: “=” and “==”
  - Example:
    - if (a = 2)
      printf("a is equal to 2 forever.\n"); // this will be printed.
      else
      printf("This statement will never be executed.\n");

Logical operator

- **&&** ----------- and
- Exp1 && exp2 is true only if both exp1 and exp2 are true
- || ------------ or
- Exp1 || exp2 is true if either exp1 or exp2 is true or if both are true
- **!** ------------not
- ! Exp1 is true if exp1 is false, and it’s false if exp1 is true

Logical operator cont.

- Examples:
  - 4<3 && 2<9
  - 5!=5 || 4<19
  - !(x<9)
- Operator precedence:
  - relational operation have higher precedence over logical operation, except the logical operation "!"
  - parentheses have the highest precedence

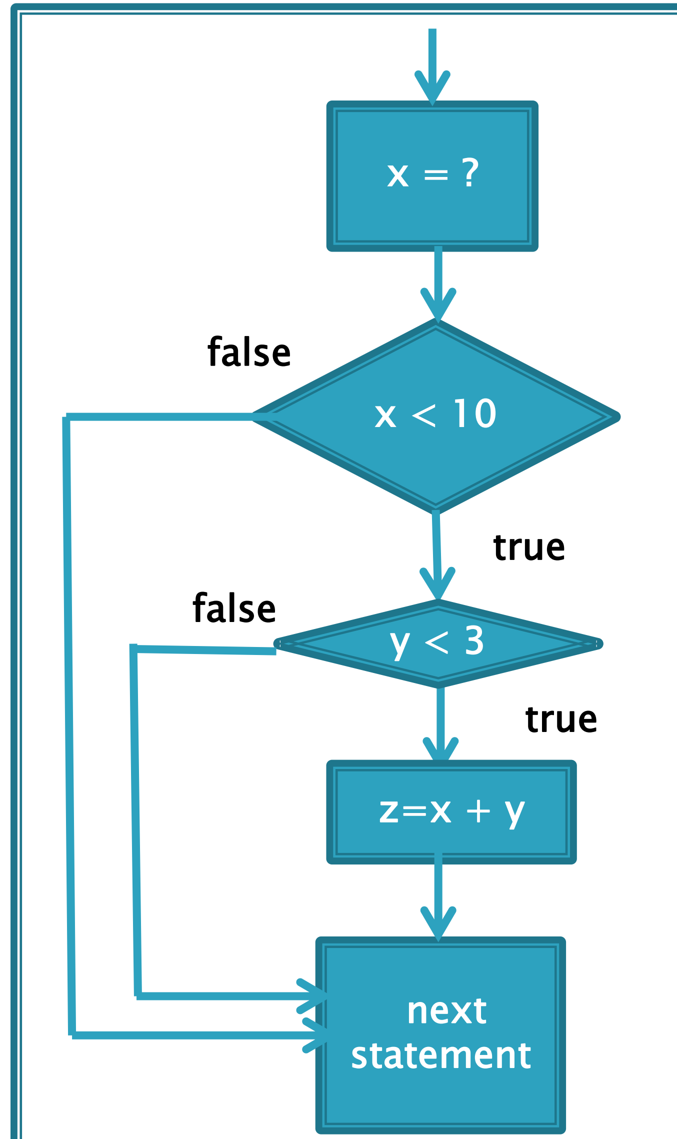
Logical operator cont.

- Example:

```
if ( x < 10 )
      if( y < 3 )
            z = x + y;
your next statement comes here;
```

➔ the same as

```
if (x < 10 && y < 3)
      z = x + y;
your next statement comes here;
```



Logical operator cont.

- To test two condition expressions, you have to use a logical operator to connect them.

```
#include <stdio.h>
int main(void)
{
	int x;
	scanf("%d",&x);
	if(3<x<6) // wrong!!!, to correct ( 3<x && x<6)
		printf("if statement\n");
	else
		printf("else statment\n");
	return 0;
}
```

- Test: input 7

Short-circuit evaluation

switch statement

- An if-else statement is used for binary decisions-those with two choices, while switch statement is intended for more than two choices.
- switch (expression)
- {
- case label1: do statements1 // there is a space between case and label
- case label2: do statements2
- .
- .
- .
- case labeln: do statementsn
- default: do defaulted statements (optional)
- }
- expression should be an integer value (including type char).
- Labels must be constants (integer constants or char constants).

- The program scans the list of labels until it finds one matching that value. Then, the program then jumps to the line.
- If there is no matching, while there is "default" key word, the statements associated with "default" will be executed.
- If there is no matching, and there is no "default" either, the program will jump out of switch statement. The statement after switch statement will be executed.

break in switch statement

- When you see “break”, the program will jump out of the switch statement when reaching the break.
- Examples:
- Example (1)
  - int x;
  - scanf("%d", &x);
  - switch(x)
  - {
  - case 1: printf("freshman\n");
  - case 2: printf("sophomore\n");
  - case 3: printf("junior\n");
  - case 4: printf("senior\n");
  - default: printf("graduates\n");
  - }
  - printf("out of switch now.\n");

- Example (2)
  - int x;
  - scanf("%d", &x);
  - switch(x)
  - {
  - case 1: printf("freshman\n");
  -             break;
  - case 2: printf("sophomore\n");
  - case 3: printf("junior\n");
  -             break;
  - case 4: printf("senior\n");
  - default: printf("graduates\n");
  - }
  - printf("out of switch now.\n");

Special cases

- Empty case:
  - int x;
  - scanf("%d", &x);
  - switch(x)
  - {
  - case 1:
  - case 2: printf("sophomore\n");
  - case 3: printf("junior\n");
  - break;
  - case 4: printf("senior\n");
  - default: printf("graduates\n");
  - }
  - printf("out of switch now.\n");
  - //It seems as if two labels are associated with one statement.

Special cases

- Case with multiple statements:
  - int x;
  - scanf("%d", &x);
  - switch(x)
  - {
  - case 1: printf("freshman\n");
  -             printf("redundant freshman\n");
  - case 2: printf("sophomore\n");
  - case 3: printf("junior\n");
  -             break;
  - case 4: printf("senior\n");
  - default: printf("graduates\n");
  - }
  - printf("out of switch now.\n");

char data type

- char type technically is an integer type
- Computer uses numeric codes to represent characters, and store characters as integers
- The mostly commonly used code in the U.S. is the ASCII code
- To read the table: Row number + column number

The ASCII character set

	0	1	2	3	4	5	6	7
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
8	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
24	CAN	EM	SUB	ESC	FS	GS	RS	US
32	SP	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7
56	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G
72	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W
88	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g
104	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL

- A char variable takes 8-bit unit of memory (1 byte), which can be verified by sizeof()
- C character constant: a single letter contained between single quotes
- Example:
  - char mych = 'a';
  - printf("%d", sizeof(char));

char data type and integer value

- char letter;
- letter = 'A';
- char letter = 'A';
- char letter = 65;
- printf("print the ASCII for \'A\' – %d", letter);// 65 will be printed
- printf("print the char value for \'A\' – %c", letter); // A will be printed
- scanf("%c", &letter);// must read a character, even the input is a digit, it will be regarded as a character
- scanf("%d", &letter);// fail – type must match
- Not good programming to mix integer and char value, because it needs remembering ASCII for characters.

Compare char values

- To compare two char values, any relational operator will work.
- Examples:
  - char ch1, ch2;
  - scanf("%c %c", &ch1, &ch2);
  - if (ch1 < ch2)
  -     printf("%c is larger.\n", ch2);
  - else
  -     printf("%c is larger.\n", ch1);

Nonprinting characters

- Characters which can not be printed directly
- Rather, some represent some actions such as backspacing or going to the next line or making the terminal bell ring.

Escape sequence	Description
\'	single quote (byte 0x27)
\"	double quote (byte 0x22)
\\	backslash (byte 0x5c)
\0	null character (byte 0x00)
\a	audible bell (byte 0x07)
\b	backspace (byte 0x08)
\f	form feed - new page (byte 0x0c)
\n	line feed - new line (byte 0x0a)
\r	carriage return (byte 0x0d)
\t	horizontal tab (byte 0x09)
\v	vertical tab (byte 0x0b)
\nnn	octal byte (nnn)
\nn	hexadecimal byte (nn)

- char newline1 = '\n';
- char newline2 = 10;
- printf("The first line. %c", newline1);
- printf("The second line. \n");
- printf("The third line. %c", newline2);