CPS 188

Computer Programming Fundamentals

Prof. Alex Ufkes

Topic 4.1: Logic and simple selection

Toronto Metropolitan University

Notice!

Obligatory copyright notice in the age of digital delivery and online classrooms:

Today

- Conditions
- Logical expressions
- Control flow
- Branching

THE

C

PROGRAMMING LANGUAGE

=

?

First thing's first: What is a condition?

An ***arithmetic*** expression evaluates to a numeric value

A ***logical*** expression evaluates to True or False.

A ***condition*** is another term for logical expression

Logical Operators

The output of a logical operator is a Boolean value.

True and **False** are Boolean values, in the same way that **5** is an integer value.

However, C does not have a Boolean data type.
In practice, we use integers: **True** = **1**, **False** = **0**.

3

Logical Operators

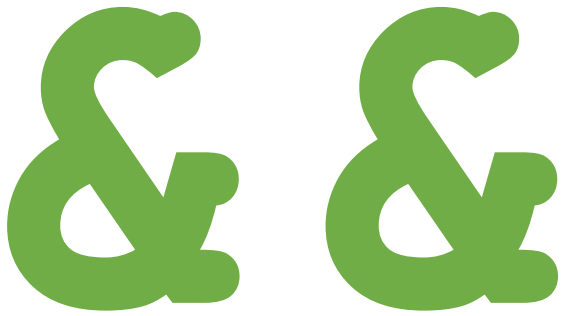
The NOT Operator

True when operand is false, false when operand is true

x	!x
1	0
0	1

The AND Operator

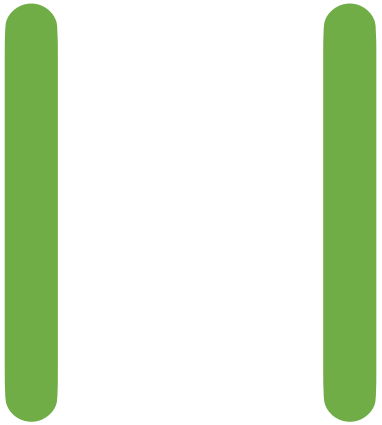
True when <u>both</u> operands are true



x	y	x && y
1	1	1
1	0	0
0	1	0
0	0	0

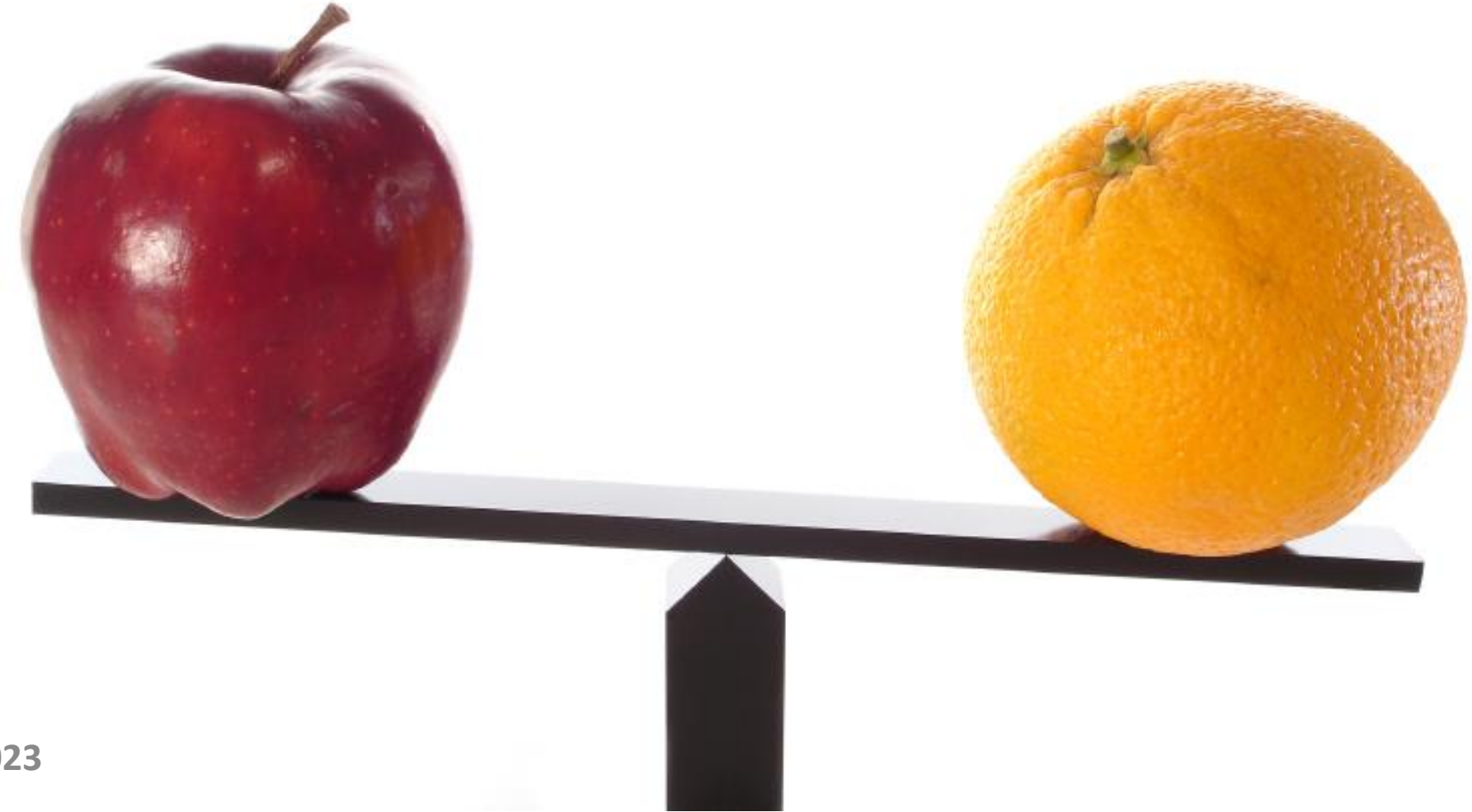
The OR Operator

True when <u>at least one</u> operand is true



x	y	x || y
1	1	1
1	0	1
0	1	1
0	0	0

Comparison Operators



6

Comparison Operators

You've seen these before…

Greater than (>)

`4 > 3`	True
`5 > 5`	False

Less than (<)

`-4 < 4`	True
`5 < 5`	False

Greater than or equal to (>=)

`3.1 >= 3.1`	True
`5 >= 6`	False

Less than or equal to (<=)

`-1 <= 0`	True
`-1 <= -2`	False

Equality (==)

`7 == 7`	True
`-1 == 1`	False

Not equal (!=)

`3 != 4`	True
`5 != 5`	False

Building

Logical expressions

logical expression

expression that contains one or more comparison and/or logical operators

```
q = c + 1 < a && c >= b*2 || -d == c
```

Operator Precedence Revisited

1. **Function calls:** `sqrt(x), log(y), ( )`
2. **Unary:** `-, +, !, (int), (double)`
3. **Binary:** `*, /, % -> +, -`
4. **Comparison:** `<, <=, >=, > -> ==, !=`
5. **Logical:** `&& -> ||`
6. **Assignment:** `=`

```
q = c + 1 < a && c >= b*2 || -d == c
```

1. **Function calls:** `sqrt(x), log(y), ( )`
2. **Unary:** `-, +, !, (int), (double)`
3. **Binary:** `*, /, % -> +, -`
4. **Comparison:** `<, <=, >=, > -> ==, !=`
5. **Logical:** `&& -> ||`
6. **Assignment:** `=`

```
q = c + 1 < a && c >= b*2 || -d == c
```

```
q = ((((c + 1) < a) && (c >= (b*2))) || ((-d) == c))
```

<u>When in doubt, use parentheses!</u>

Numeric Operands & Logical Operators

- It is perfectly legal to use numeric values with logical operators.
- ***Just don't do it by accident.***

<u>Numeric operands with logical operators:</u>

- In C, if the number is 0, it is considered **FALSE.**
- If the number is **ANYTHING** other than 0, it is considered **TRUE**

```
   357 && (1 < 5)     TRUE (1)
-12.77 && (3 - 3)    FALSE (0)
```

Logical Expression Output

Quincy 2005 - [Text1]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main (void)
{
    int x = 357 && (1 < 5);
    int y = -12.77 && (3 - 3);

    printf("x = %d\n", x);
    printf("y = %d\n", y);

    return 0;
}
```

Press F1 for help Ln 6, Col 27

Quincy 2005

```
x = 1
y = 0

Press Enter to return to
 Quincy...
```

Provide a logical expression for the following:

Are **x** and **y** less than 5?

```
(x < 5) && (y < 5)
```

```
(x < 5) && (y < 5)
```

Are these the same?

~~`x && y < 5`~~

Computers + Intuition

Expressions must be built **<u>unambiguously</u>**. The compiler doesn't understand what you ***meant***, it does ***precisely*** what you tell it.

```
x && y < 5
```

Are `x` and `y` less than 5?

```
(x < 5) && (y < 5)
```

Is **x** equal to 1 or 3?

```
(x == 1) || (x == 3)
```

Is **x** between 3 and 5?

```
(x >= 3) && (x <= 5)
```

```
(x >= 3) && (x <= 5)
```

Are these the same?

~~3 <= x <= 5~~

NO!

```
3 <= x <= 5
```

This will evaluate first and will be either **true** or **false**.

```
true <= 5?      false <= 5?

1 <= 5?         0 <= 5?
```

Always **TRUE**!

Is x an odd number?

Trickier. What is an odd number?

- Any number not evenly divisible by 2.
- How can we check if some number x is divisible by 2?

Remember the remainder (%) operator?

```
x % 2 != 0
```

Is x an even number?

```
x % 2 == 0
```

Is x an odd number?

```
x % 2 != 0
```

-or-

```
x % 2 == 1
```

TRUE OR FALSE

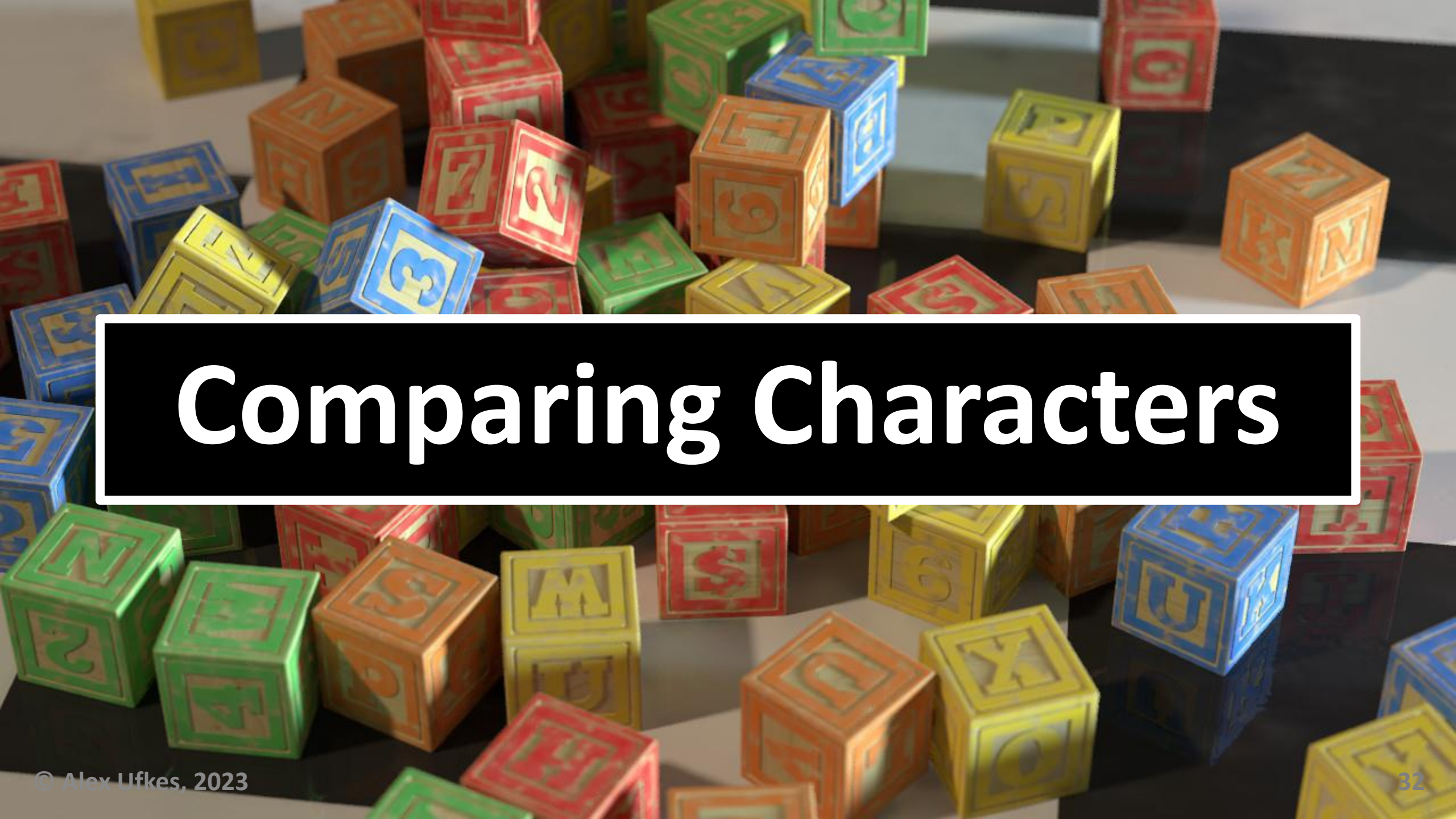
```
a = 1
b = 2
c = 3
```

```
(a<5-3) && (b==5-c) && (b>4)
```

TRUE or FALSE?

```
0 && -0.00001 || 357
```

TRUE or FALSE?



Comparing Characters

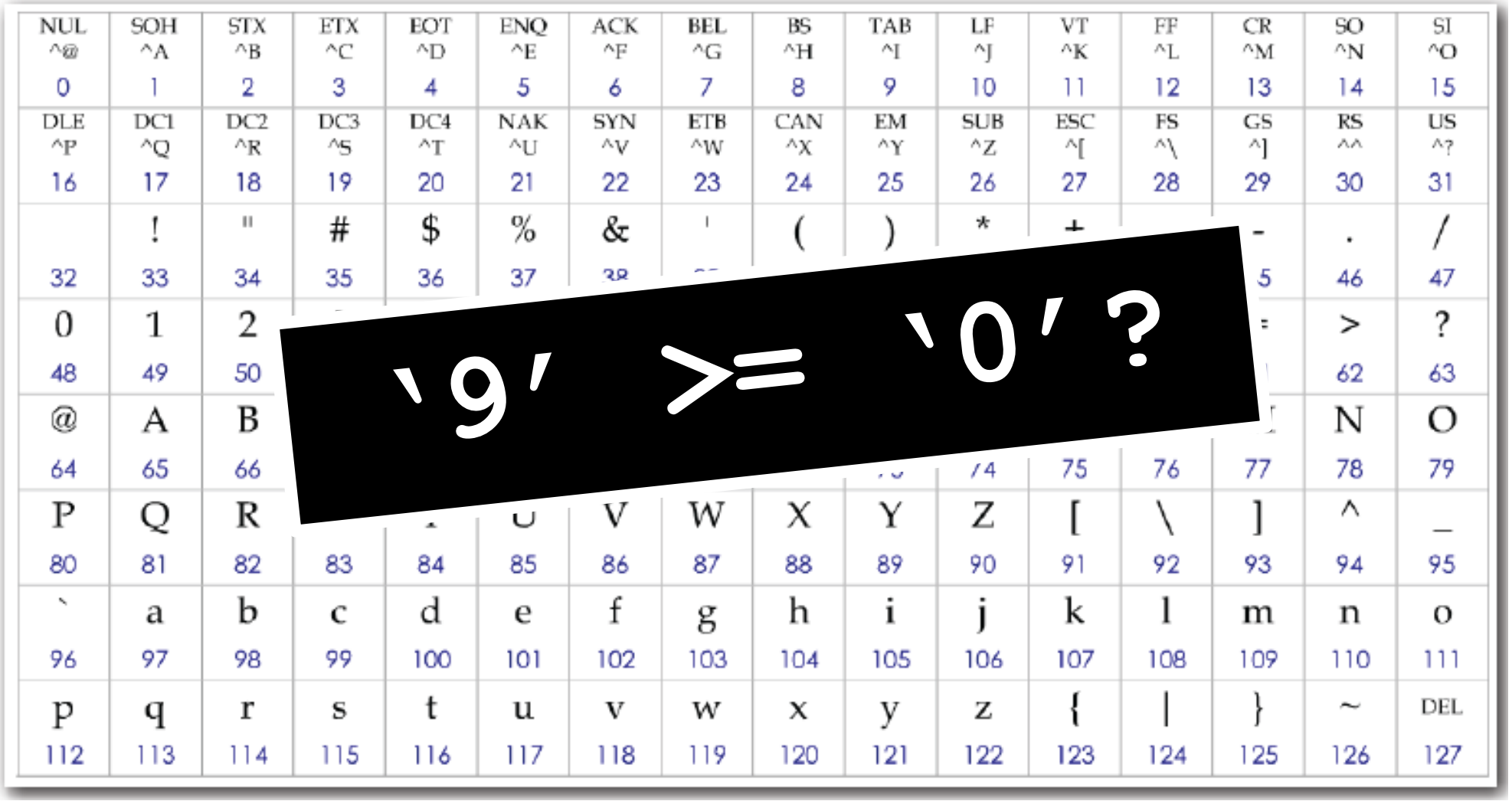
```
'9' >= '0'?

'a' < 'e'?

'B' <= 'A'?

'Z' == 'z'?
```

How does this work?

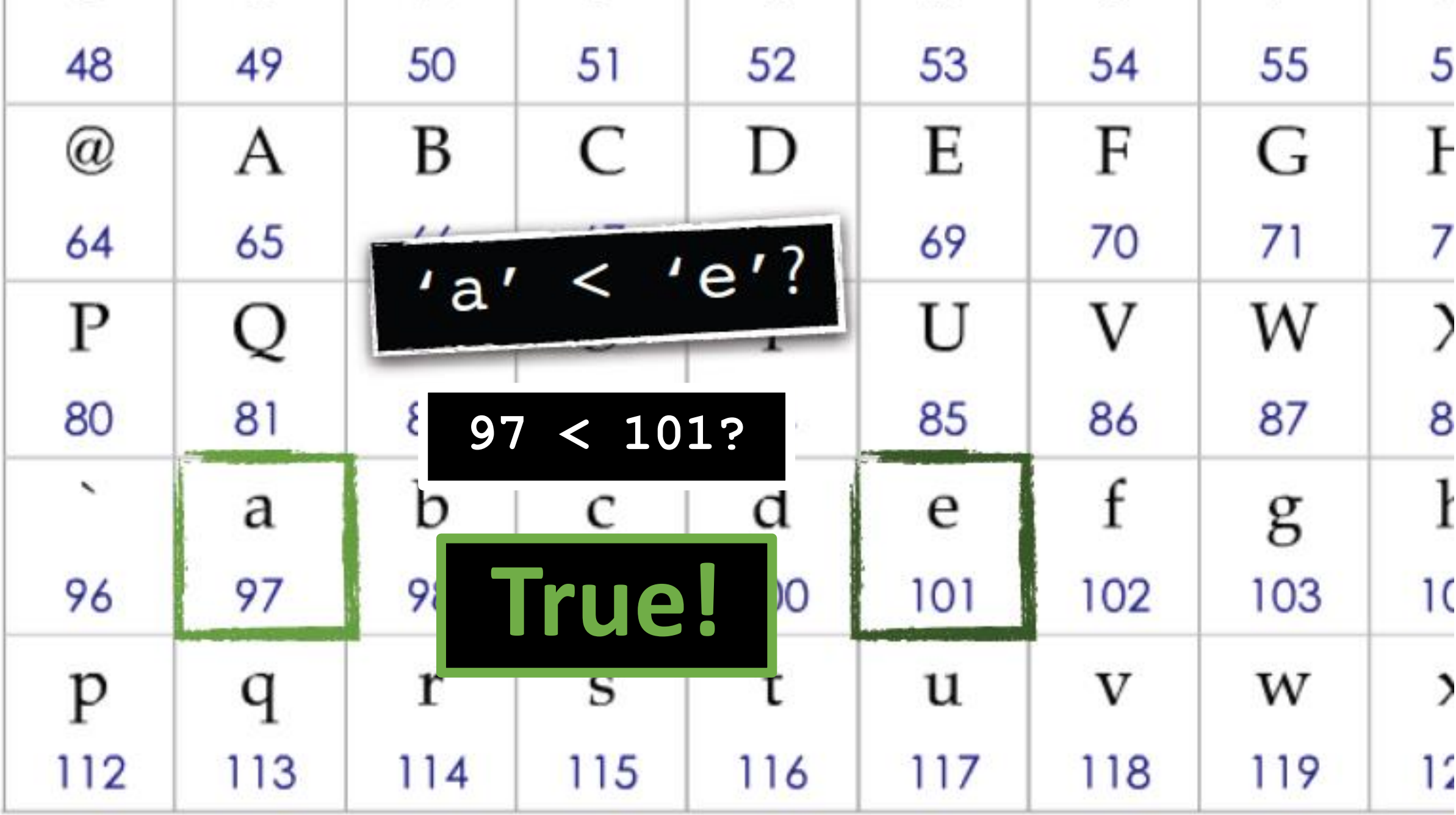

'9' >= '0'?

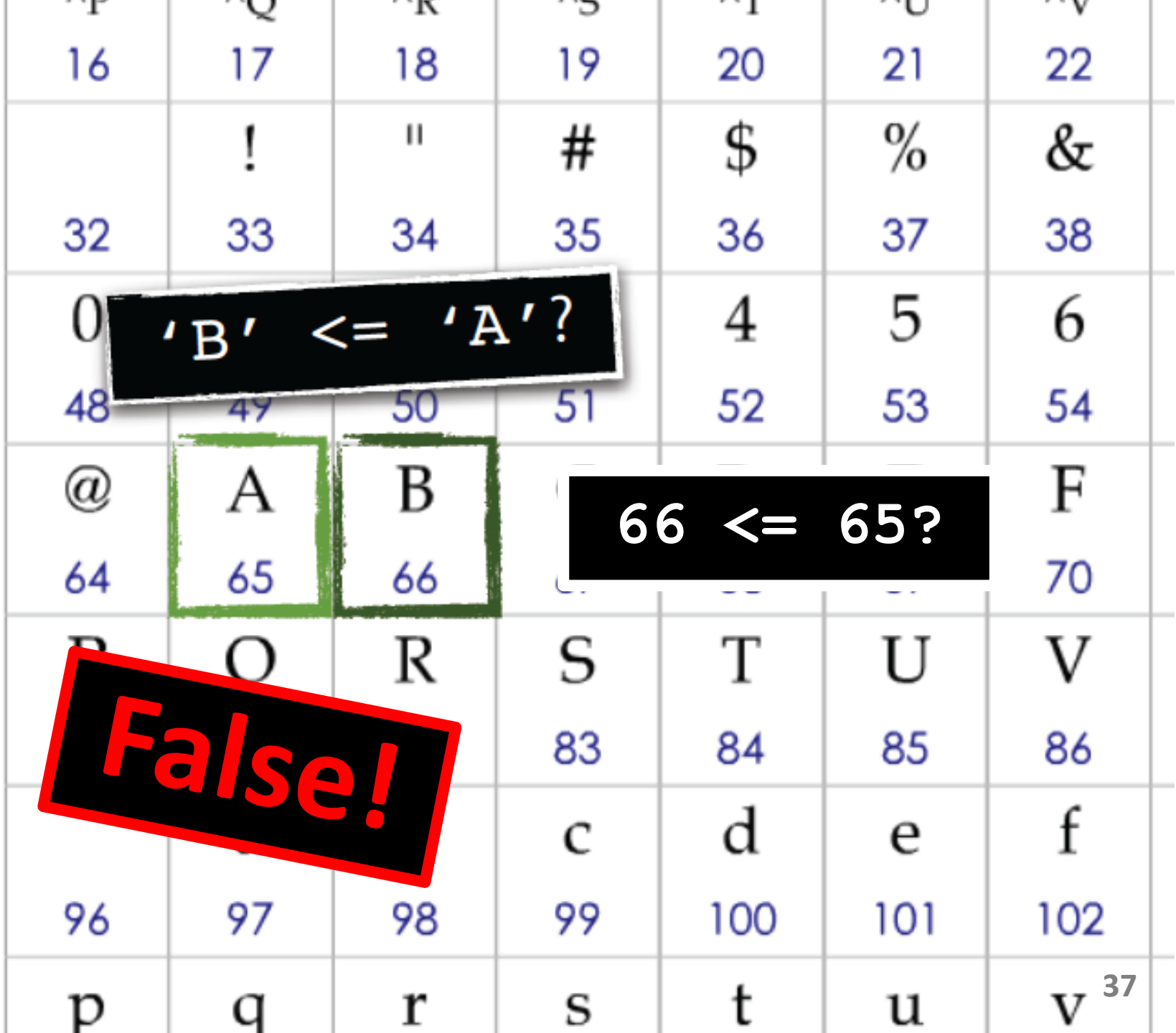
NUL ^@ 0	SOH ^A 1	STX ^B 2	ETX ^C 3	EOT ^D 4	ENQ ^E 5	ACK ^F 6	BEL ^G 7	BS ^H 8	TAB ^I 9	LF ^J 10
DLE ^P 16	DC1 ^Q 17	DC2 ^R 18	DC3 ^S 19	DC4 ^T	NAK ^U	SYN ^V 22	ETB ^W 23	CAN ^X 24	EM ^Y 25	SUB ^Z 26
32	! 33	" 34	# 35	36	37	& 38	' 39	(40	) 41	* 42
0 48	1 49	2 50	51	52	53	54	7 55	8 56	9 57	: 58
@ 64	A 65	B 66	C 67				G 71	H 72	I 73	J 74
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90

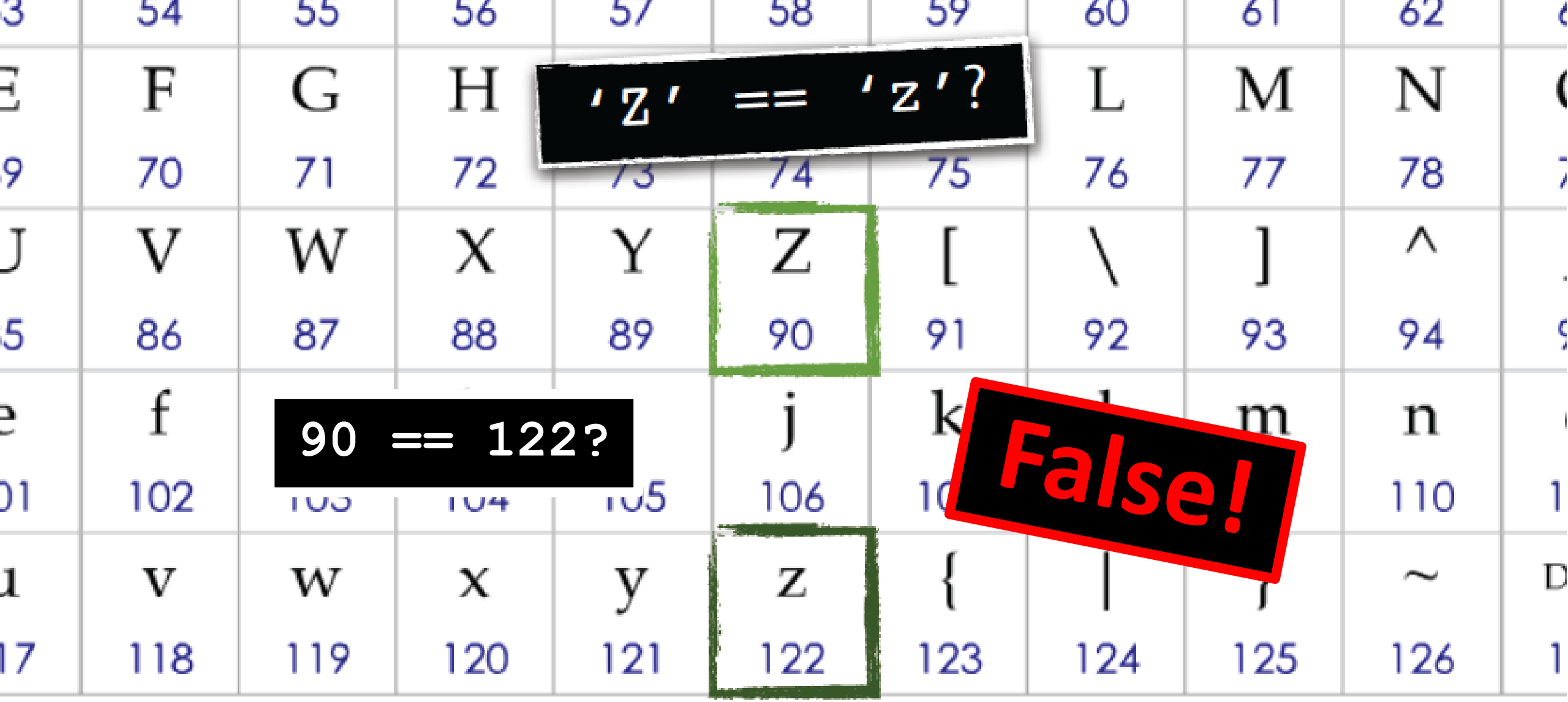
True!

'9' >= '0'?

57 >= 48?






'Z' == 'z'?
90 == 122?
False!

NUL ^@ 0	SOH ^A 1	STX ^B 2	ETX ^C 3	EOT ^D 4	ENQ ^E 5	ACK ^F 6	BEL ^G 7	BS ^H 8	TAB ^I 9	LF ^J 10	VT ^K 11	FF ^L 12	CR ^M 13	SO ^N 14	SI ^O 15
DLE ^P 16	DC1 ^Q 17	DC2 ^R 18	DC3 ^S 19	DC4 ^T 20	NAK ^U 21	SYN ^V 22	ETB ^W 23	CAN ^X 24	EM ^Y 25	SUB ^Z 26	ESC ^[27	FS ^\ 28	GS ^] 29	RS ^^ 30	US ^? 31
32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

Is the variable `ch` lowercase?

Is the variable ch lowercase?

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

```
(ch >= 'a') && (ch <= 'z')
```

Is the variable `ch` a letter?

NUL ^@ 0	SOH ^A 1	STX ^B 2	ETX ^C 3	EOT ^D 4	ENQ ^E 5	ACK ^F 6	BEL ^G 7	BS ^H 8	TAB ^I 9	LF ^J 10	VT ^K 11	FF ^L 12	CR ^M 13	SO ^N 14	SI ^O 15
DLE ^P 16	DC1 ^Q 17	DC2 ^R 18	DC3 ^S 19	DC4 ^T 20	NAK ^U 21	SYN ^V 22	ETB ^W 23	CAN ^X 24	EM ^Y 25	SUB ^Z 26	ESC ^[27	FS ^\ 28	GS ^] 29	RS ^^ 30	US ^? 31
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

Is the variable ch a letter?

48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

```
((ch >= ‘a’) && (ch <= ‘z’)) ||
((ch >= ‘A’) && (ch <= ‘Z’))
```

Floating-Point Equality

Quincy 2005 - [test2.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main (void)
{
    double x = 0.1;
    double y = 0.3;
    double z = 0.1 + 0.1 + 0.1;

    printf("%.21f\n", x);
    printf("%.21f\n", y);
    printf("%.21f\n", z);

    printf("%d\n", y == z);

    return (0);
}
```

Press F1 for help | Ln 18, Col 1 | N

Be careful! Floating-point representation is ***not precise!***

Floating-Point Equality

Quincy 2005 - [test2.c]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>

int main (void)
{
    double x = 0.1;
    double y = 0.3;
    double z = 0.1 + 0.1 + 0.1;

    printf("%.21f\n", x);
    printf("%.21f\n", y);
    printf("%.21f\n", z);

    printf("%d\n", y == z);

    return (0);
}
```

Press F1 for help Ln 18, Col 1 N

quincy

```
0.100000000000000010000
0.299999999999999990000
0.300000000000000040000
0

Press Enter to return to Quincy...
```

Control Structures

Control Structures

Determine the sequence of execution of a set of instructions.

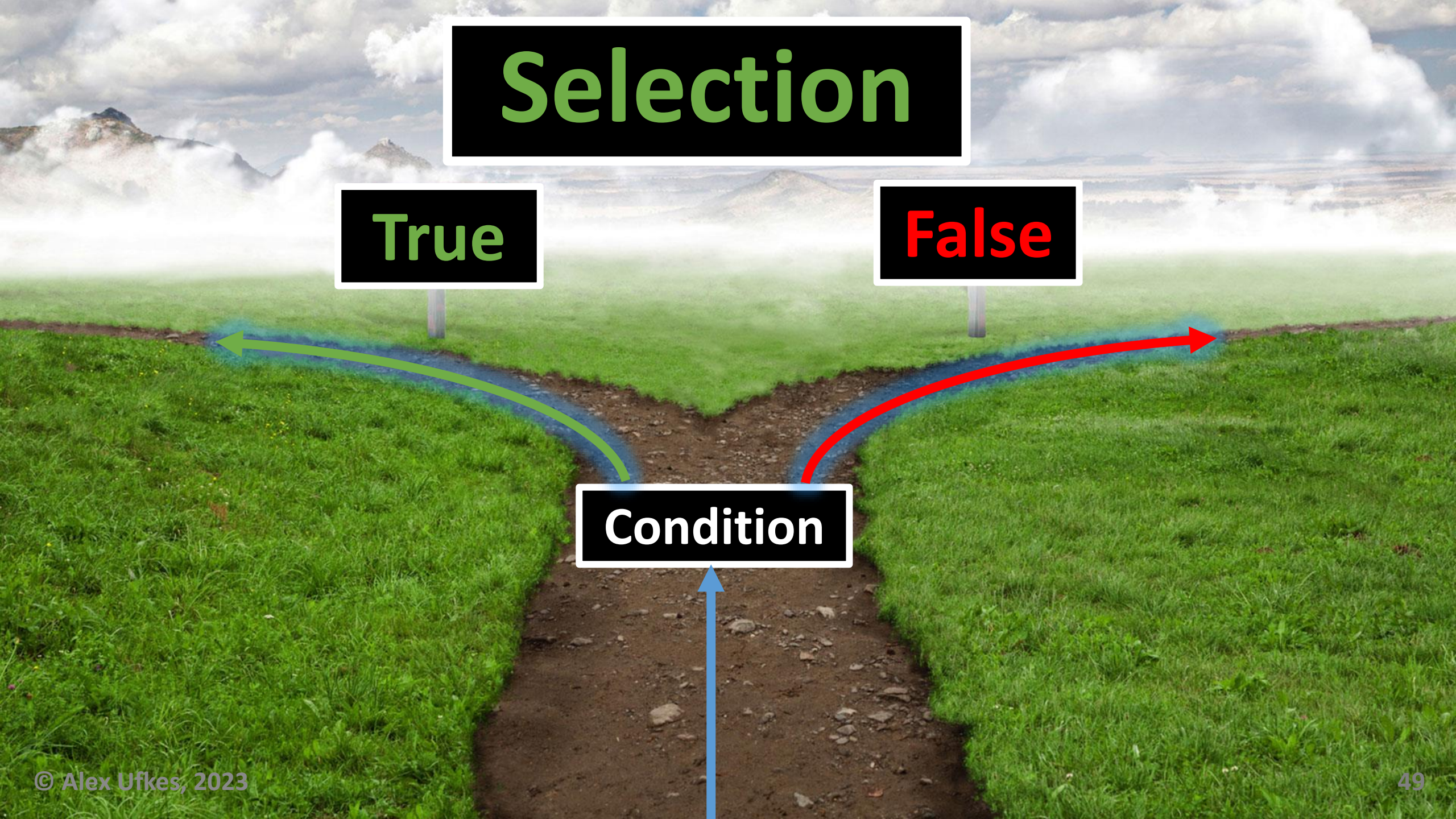
Sequence

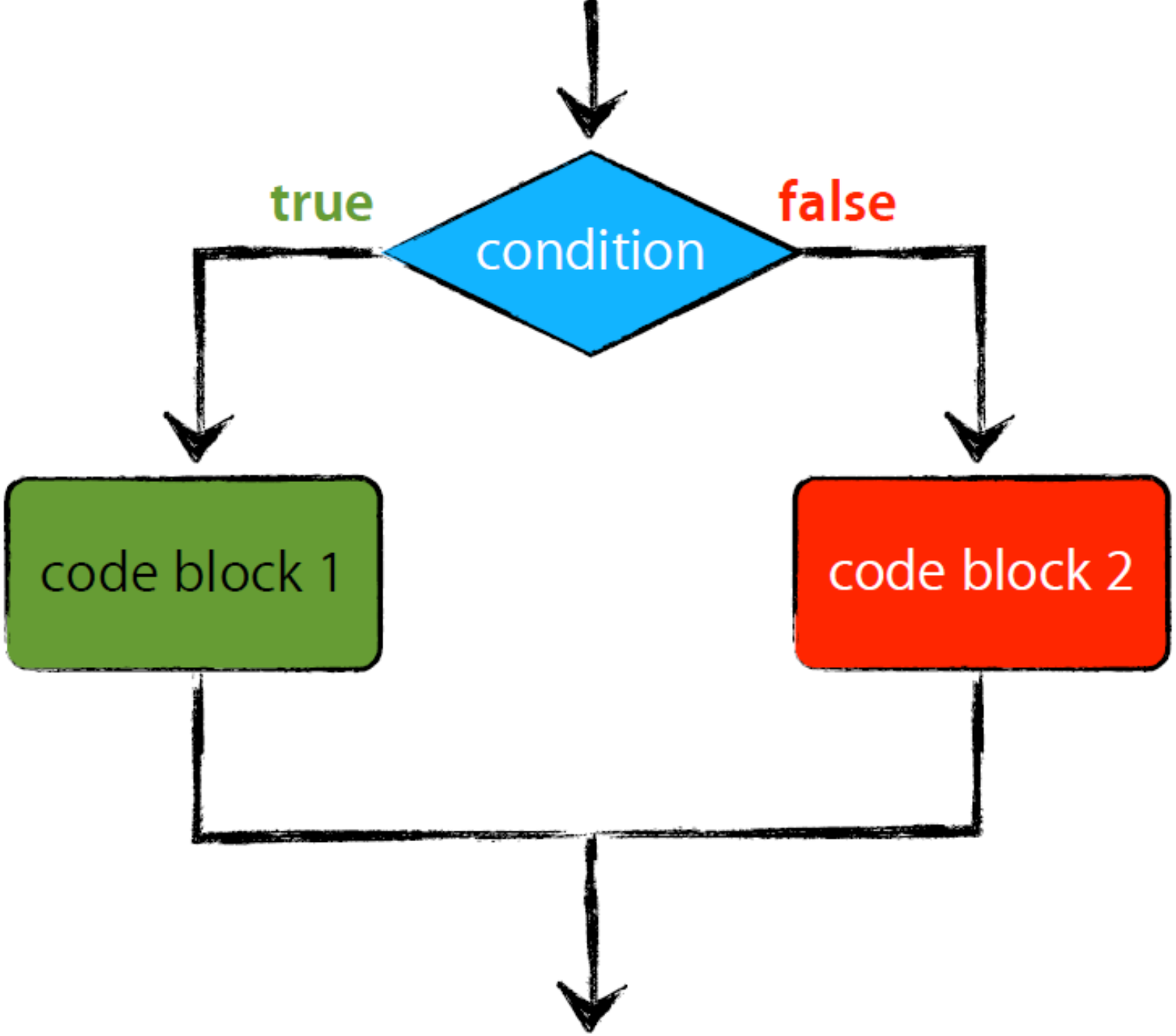
Everything we've seen so far has been sequential

```
int x;
```

```
x = 255;
```

```
printf("%d\n", x);
```





The `if` Statement

```
/* single branch */

if (condition)
    condition is true, execute statement;
```

Single Statements

The `if` only applies to a <u>single</u> statement.

```
int x;
scanf("%d", &x);
if (x > 0)
  printf("Prints if x > 0! \n");
printf("Prints no matter what! \n");
```

This ***`printf`*** *is not part of the* ***`if`*** *statement*

Compound Statements

```
int x;
scanf("%d", &x);

if (x > 0)
{
  printf("Prints if x > 0! \n");
  printf("Prints if x > 0! \n");
}
```

Compound statement

We can include as much (or as little) code within the `if` as we want

Pro Tip

Always use curly braces **{ }**, even for single statements

```
int x;
scanf("%d", &x);

if (x > 0)
{
    printf("Prints if x > 0! \n");
}
```

```
int temp;

printf("What is the temperature? ");
scanf("%d", &temp);

if (temp >= 100)
{
  printf("The water is boiling!\n");
}
```

Input:	Output:
107	The water is boiling!
24	

```
int temp;

printf("What is the temperature? ");
scanf("%d", &temp);

if (temp >= 100)
{
  printf("The water is boiling!\n");
}
```

NO semicolon!

never put a semicolon after the condition!

Why?

never put a semicolon after the condition!

Recall: Semicolons are used to **end** a statement.

```
if (temp >= 100);  /* ends the if statement! */
{
  printf("The water is boiling!\n");
}
```

Written like this, `printf` will always execute!

Once the `if` statement is ended, as above, the `printf` is no longer part of the `if`, and executes in sequence as usual.

Two Branches

```
if (condition)
  condition is true, execute statement;
else
  condition is false, execute statement;
```

```
int temp;

printf("What is the temperature? ");
scanf("%d", &temp);

if (temp >= 20) {
   printf("It is warm outside \n"); }
else {
   printf("It is cool outside \n"); }
```

Input:	Output:
17	It is cool outside
24	It is warm outside

Quincy 2005 - [test *]

File Edit View Project Debug Tools Window Help

```c
#include <stdio.h>
int main (void)
{
    int temp;

    printf("What is the temperature? ");
    scanf("%d", &temp);

    if (temp >= 20)
    {
        printf("It is warm outside \n");
    }
    else
    {
        printf("It is cool outside \n");
    }
}
```

Press F1 for help | Ln 6, Col 38 | NUM

Quincy 2005

```
What is the temperature? 17
It is cool outside

Press Enter to return to Quincy...
```

Quincy 2005

```
What is the temperature? 24
It is warm outside

Press Enter to return to Quincy...
```

In Summary

- Logical & comparison operators
- Logical expressions & conditions
- Branching with **`if/else`**

Questions?