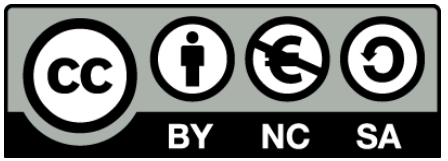
Strings

UE 101: Algorithms and Programming

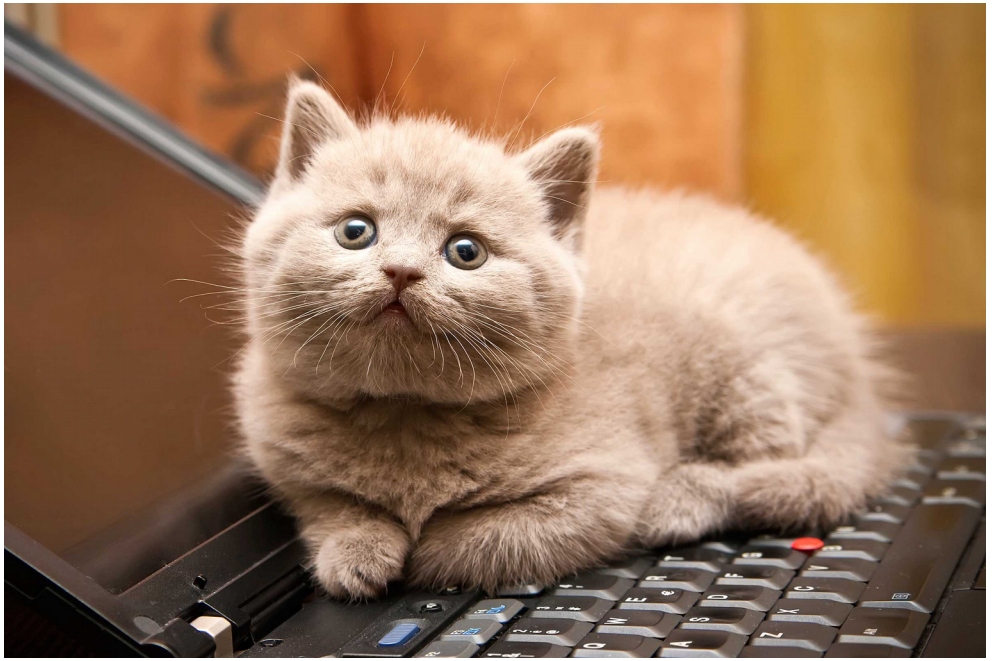
Soham Pal



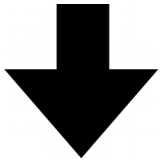
Inspired by past offerings of **CS50**.

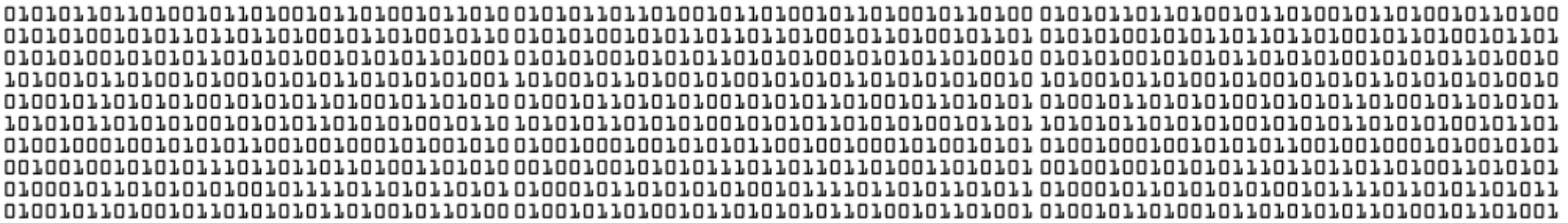
CS50 2011-2016 by **David J. Malan** of **Harvard University**.

Data Representation

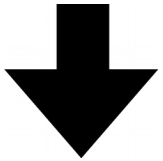


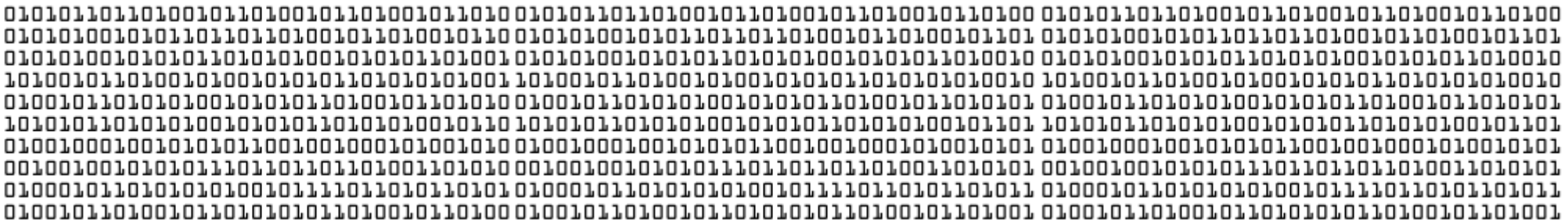

AUDIO
VIDEO






TXT





Types

Type	Storage	Device-dependent?
int	4 bytes = 32 bits	✓
float	4 bytes = 32 bits	✓
double	8 bytes = 64 bits	✓
char	1 byte = 8 bits	✘

Numbers

$(123)_{10}$

Numbers

$(123)_{10} = 3 + 20 + 100$

Numbers

$$(123)_{10} = 3 + 2\times10^{1} + 1\times10^{2}$$

Integers

$$(1010)_2 = 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$

Integers

$$(1010)_2 = 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$

Integers

$$(1010)_2 = 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 0 + 2 + 0 + 8$$

Integers

$$
\begin{aligned}
(1010)_2 &= 0 + 1\times2^1 + 0\times2^2 + 1\times2^3 \\
&= 0 + 1\times2^1 + 0\times2^2 + 1\times2^3 \\
&= 0 + 2 + 0 + 8 \\
&= (10)_{10}
\end{aligned}
$$

Integers

$$(1010)_2 = 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 0 + 1\times2^1 + 0\times2^2 + 1\times2^3$$
$$= 0 + 2 + 0 + 8$$
$$= (10)_{10}$$

int x = 10;

```
0000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

Negative Integers

int x = 10;

```
0000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

int x = -10;

Negative Integers

int x = 10;

```
0000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

int x = -10;

```
1000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

Negative Integers

int x = 10;

```
0000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

int x = -10;

```
1111 1111  1111 1111  1111 1111  1111 1111
1111 1111  1111 1111  1111 1111  1111 0101
```

Negative Integers

int x = 10;

```
0000 0000  0000 0000  0000 0000  0000 0000
0000 0000  0000 0000  0000 0000  0000 1010
```

int x = -10;

```
1111 1111  1111 1111  1111 1111  1111 1111
1111 1111  1111 1111  1111 1111  1111 0110
```

Floats

$(0.15625)_{10}$

Floats

$(0.15625)_{10} = (0.00101)_2$

Floats

$$(0.15625)_{10} = (0.00101)_2$$
$$= (1.01)_2 \times 2^{-3}$$

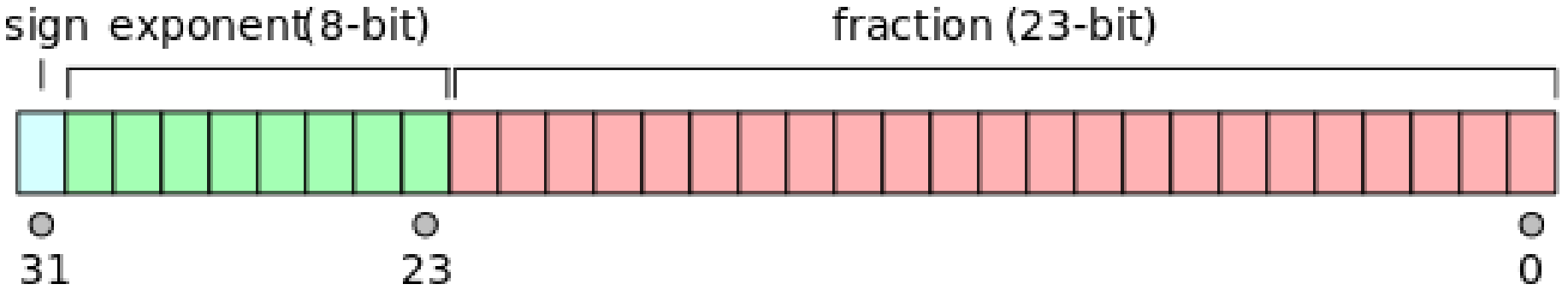
Floats

$$(0.15625)_{10} = (0.00101)_2$$
$$= (1.01)_2 \times 2^{-3}$$
$$= (1.01 \times 10^{-11})_2$$

Floats

$$(0.15625)_{10} = (0.00101)_2$$
$$= (1.01)_2 \times 2^{-3}$$
$$= (1.01 \times 10^{-11})_2$$

+ exponent = -11 **fraction = .01**



Video

Floats

$$(0.2)_{10} = (0.0011\ 0011\ 0011\ \ldots)_2$$

Characters

```
char ch;

scanf  ("%c", &ch);
printf ("%c", ch);
```

Characters

A	B	C	D	E	F	G	...
65	66	67	68	69	70	71	...

a	b	c	d	e	f	g	...
97	98	99	100	101	102	103	...

Characters

```
char ch;


ch = 'x';
printf ("%c", ch);


// Prints: x
```

Characters

```
char ch;


ch = 'x';
printf ("%d", ch);
```

Characters

```
char ch;


ch = 'x';
printf ("%d", ch);


// Prints: 120
```

Characters

A	B	C	D	E	F	G	...
65	66	67	68	69	70	71	...

a	b	c	d	e	f	g	...
97	98	99	100	101	102	103	...

0	1	2	3	4	5	6	...
48	49	50	51	52	53	54	...

Text

```
string my_string;

my_string = "Subaru";
```

Text

```
string my_string;

my_string = "Subaru";
```

S	u	b	a	r	u

Text

```
char* my_string;

my_string = "Subaru";
```

S	u	b	a	r	u

Text

```
for (i=0; i<6; i++)
{
    printf ("%c", my_string[i]);
}
```

S	u	b	a	r	u

Text

```
for (i=0; i<strlen(my_string); i++)
{
    printf ("%c", my_string[i]);
}
```

S	u	b	a	r	u

Text

```
n = strlen(my_string);


for (i=0; i<n; i++)
    printf ("%c", my_string[i]);
```

S	u	b	a	r	u

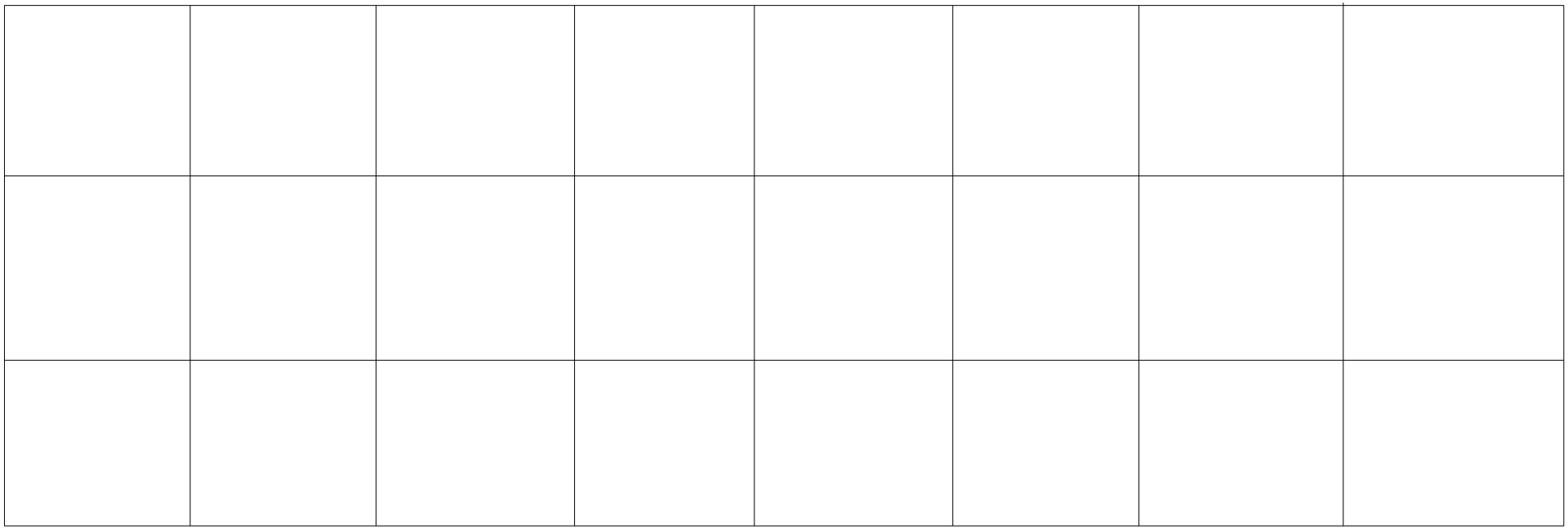
Text

```
char* my_string = GetString();
int n = strlen(my_string);


for (int i=0; i<n; i++)
    printf ("%c", my_string[i]);
```

Text

```
char* name1 = "Subaru";
char* name2 = "Emilia";
char* name3 = "Wilhelm";
```



Text

```
char* name1 = "Subaru";
char* name2 = "Emilia";
char* name3 = "Wilhelm";
```

S	u	b	a	r	u		E
m	i	l	i	a		W	i
l	h	e	l	m			

Text

```
char* name1 = "Subaru";
char* name2 = "Emilia";
char* name3 = "Wilhelm";
```

S	u	b	a	r	u	⊥	E
m	i	l	i	a	⊥	W	i
l	h	e	l	m	⊥		

Text

```
char* name1 = "Subaru";
char* name2 = "Emilia";
char* name3 = "Wilhelm";
```

S	u	b	a	r	u	\0	E
m	i	l	i	a	\0	W	i
l	h	e	l	m	\0		

Text

```
char* my_string = GetString();
int n = strlen(my_string);


for (int i=0; i<n; i++)
    printf ("%c", my_string[i]);
```

Text

```
char* my_string = GetString();
int n = strlen(my_string);


printf ("%s", my_string);
```

Length of a String

String Length

```
char* my_string = GetString();
int i = 0;


while (my_string[i] != '\0')
    i++;
```

Capitalization

Captialization

```
char ch;
```

Characters

A	B	C	D	E	F	G	...
65	66	67	68	69	70	71	...

a	b	c	d	e	f	g	...
97	98	99	100	101	102	103	...

0	1	2	3	4	5	6	...
48	49	50	51	52	53	54	...

ASCII

American Standard Code for Information Interchange

Characters

A	B	C	D	E	F	G	...
65	66	67	68	69	70	71	...

a	b	c	d	e	f	g	...
97	98	99	100	101	102	103	...

0	1	2	3	4	5	6	...
48	49	50	51	52	53	54	...

Captialization

```
char ch;


// Capitalizing a single letter
if (letter >= 97 && letter <= 122)
    letter -= (97-35);
```

Captialization

```
char ch;


// Capitalizing a single letter
if (letter >= 97 && letter <= 122)
    letter -= 32;
```

Captialization

```
char ch;


// Capitalizing a single letter
if (letter >= 'a' && letter <= 'z')
    letter -= 'a' - 'A';
```

Captialization

```
#include <ctype.h>


// Capitalizing a single letter
if (islower(letter))
    letter -= toupper(letter);
```

Captialization

```
for (int i = 0; i < strlen(str); i++)
{
    if(islower(str[i]))
        str[i] = toupper(str[i]);
}
```

Lexicographic Order

s1 < s2
“alpha” < “beta”

s1 > s2
“pizza” > “pancake”

s1 == s2
“test” == “test”

Comparing Strings

```
char* s1 = GetString();
char* s2 = GetString();

int result = 0;

for (i = 0; i <= n && result == 0; i++) {
    result = str1[i] - str2[i];
    i++;
}
```

Comparing Strings

```
char* s1 = GetString(); char* s2 = GetString();
int n1 = strlen(s1);    int n2 = strlen(s2);

int n = min(n1, n2);
int result = 0;

for (i = 0; i <= n && result == 0; i++) {
    result = str1[i] - str2[i];
    i++;
}
```

string.h

```
strcpy(destination, source);
// Copies source into destination.


strcat(destination, source);
// Appends source to destination.
// destination := destination || source;


strcmp(string_a, string_b);
// Returns lexicographic order.
// < 0, > 0 or == 0
```

Safety

```
char destination[5];
char *source = "This is a string";

// Dangerous!
strcpy(destination, source);
```

Safety

```
char *source = "This is a string";


int n = strlen(source);
char *destination = malloc(n);


strcpy(destination, source);
```

Safety

```
char *source = "This is a string";

// Ignores storage for \0!
int n = strlen(source);
char *destination = malloc(n);

strcpy(destination, source);
```

Safety

```
char *source = "This is a string";


int n = strlen(source);
char *destination = malloc(n+1);


strcpy(destination, source);
```

string.h

```
strncpy(destination, source, n);
// Copies source into destination.


strncat(destination, source, n);
// Appends source to destination.
// destination := destination || source;


strncmp(string_a, string_b, n);
// Returns lexicographic order.
// < 0, > 0 or == 0
```

string.h

```
// Searches haystack for needle.
strstr(haystack, needle);
```

string.h

```
haystack = "haystack";
needle   = "barn";

char *result = strstr(haystack, needle);
```

.

string.h

```
haystack = "haystack";
needle   = "barn";

char *result = strstr(haystack, needle);
// result is a NULL pointer.
```

.

string.h

```
haystack = "facebook";
needle   = "ebook";

char *result = strstr(haystack, needle);
```

.

string.h

```
haystack = "facebook";
needle   = "ebook";

char *result = strstr(haystack, needle);
// (result - haystack) is 3.
```

.

Reading input

- `GetString()` okay on UE101 tool.
- Real world applications:

```
char* variable = malloc(length);
fgets(variable, length, stdin);
```

Reading input

- `GetString()` okay on UE101 tool.
- Real world applications:

```
char* variable = malloc(length);
fgets(variable, length, stdin);
```

- Problem:
  - Newline gets copied as part of the input.

Reading input

- Real world applications:

```
char* variable = malloc(length);
fgets(variable, length, stdin);
variable[strlen(variable) - 1] = '\0';
```

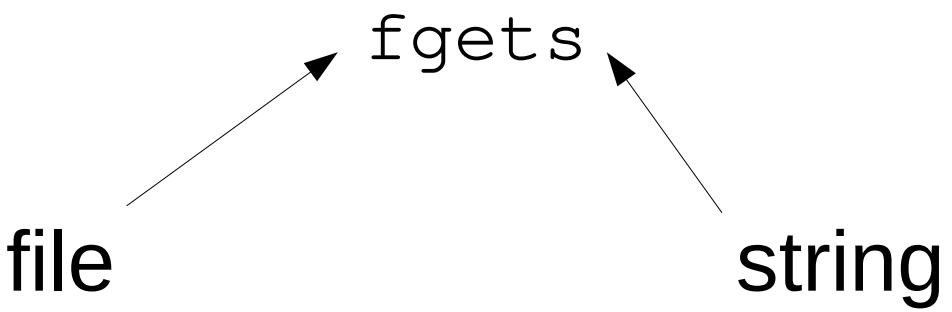
Reading input

- Real world applications:

```
char* variable = malloc(length+1);
fgets(variable, length+1, stdin);
variable[strlen(variable) - 1] = '\0';
```

Reading input

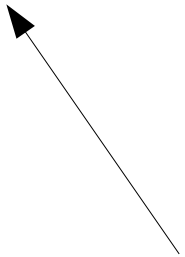
- `fgets` is defined in `stdio.h`.
- `stdin` is defined in `stdio.h`.
  - It is a pointer to a special *file* that contains characters entered by the user.



```
printf("%d %d", x, y);
  scanf("%d %d", &z);
```

Formatted I/O

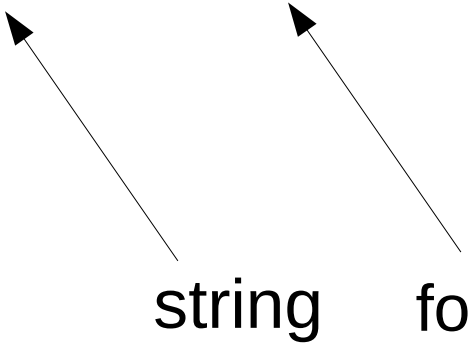
```
printf(format_string, var1, var2, ...);
```



formatted

String I/O

```
sprintf(buffer, format_string, var1, var2, ...);
```



String I/O

```
char result[100];
int a = 3;
int b = 4;

sprintf(result, "%d + %d = %d", a, b, a+b);
```

String I/O

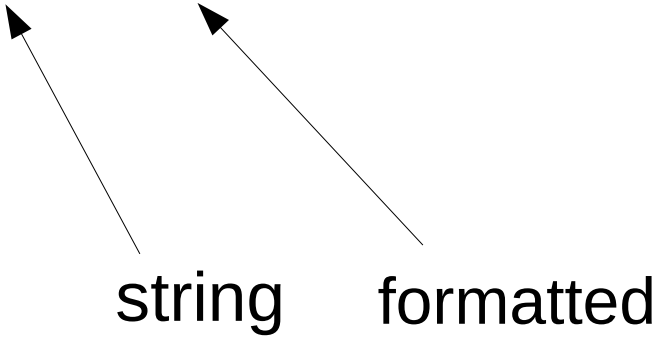
```
char result[100];
int a = 3;
int b = 4;

sprintf(result, "%d + %d = %d", a, b, a+b);

// result holds "3 + 4 = 7"
```

String I/O

```
sscanf(buffer, format_string, var1, var2, ...);
```



String I/O

```
char* source = "5 3.14 X";
int a;
float b;
char c;

sprintf(source, "%d %f %x", &a, &b, &c);
```

Characters

A	B	C	D	E	F	G	...
65	66	67	68	69	70	71	...

a	b	c	d	e	f	g	...
97	98	99	100	101	102	103	...

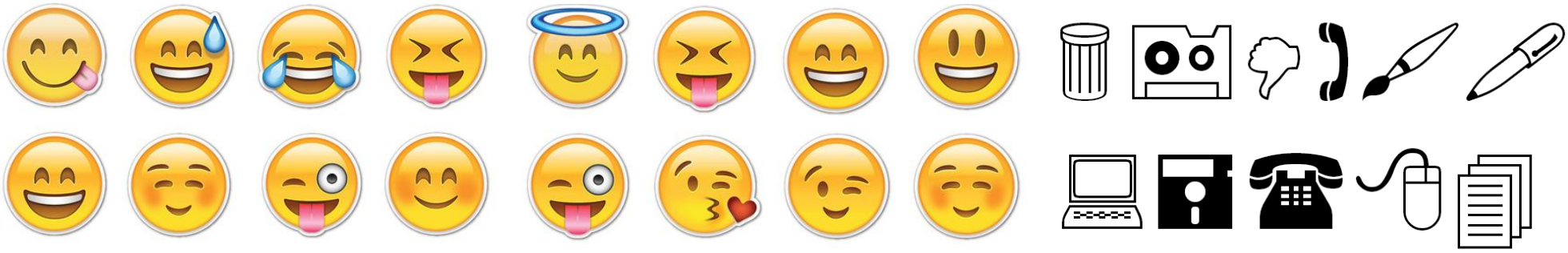
0	1	2	3	4	5	6	...
48	49	50	51	52	53	54	...

ASCII

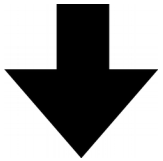
American Standard Code for Information Interchange

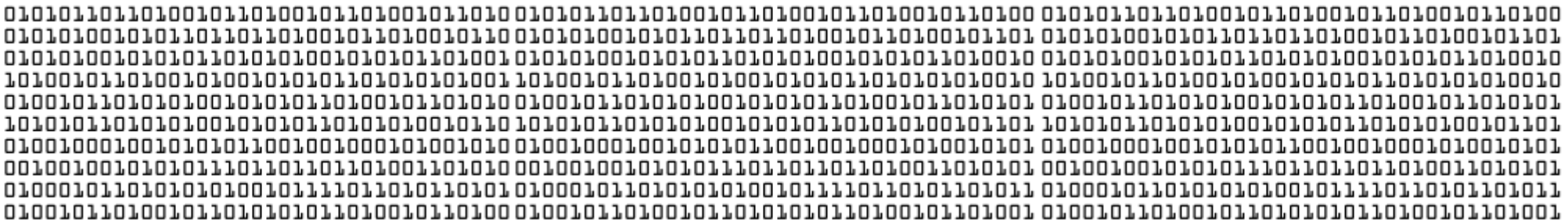
Unicode

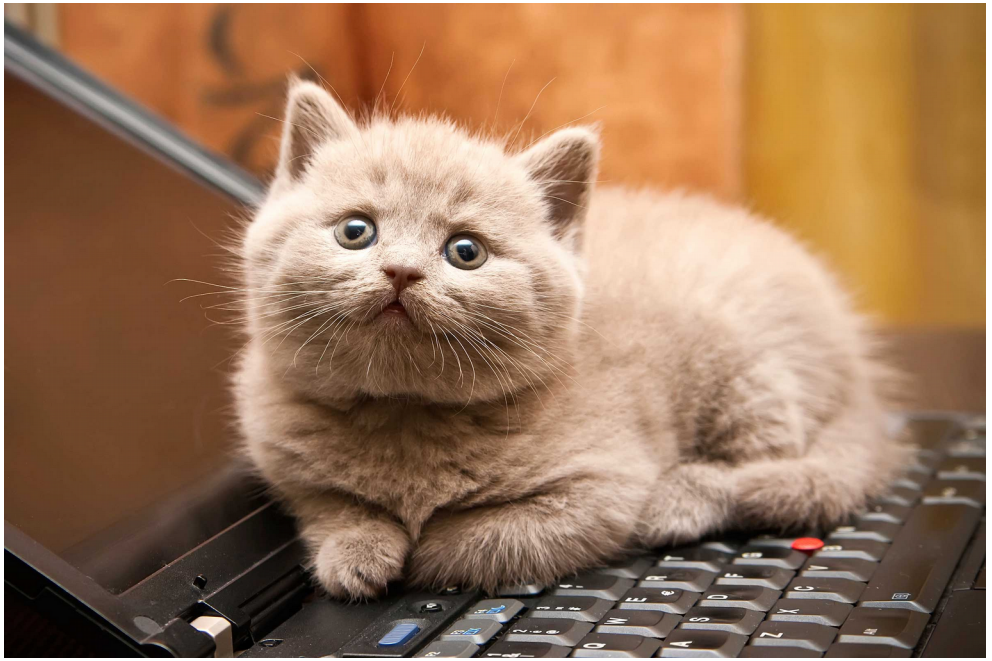
Backwards compatible (sort of)



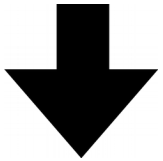

TXT

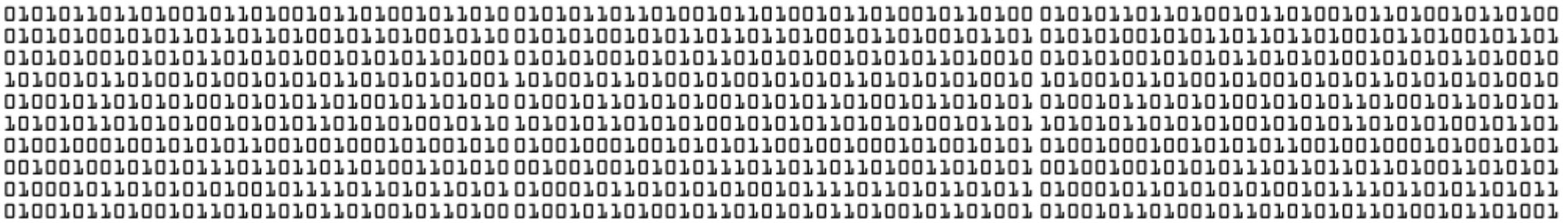






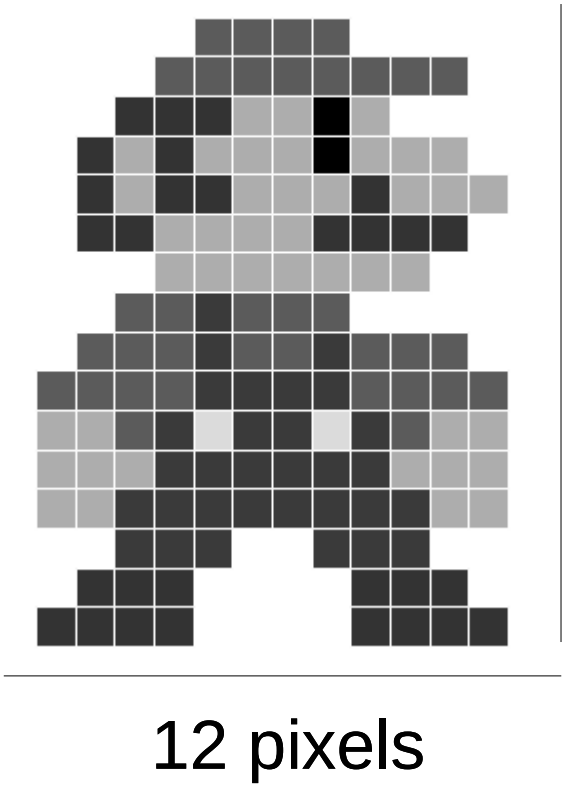

AUDIO
VIDEO





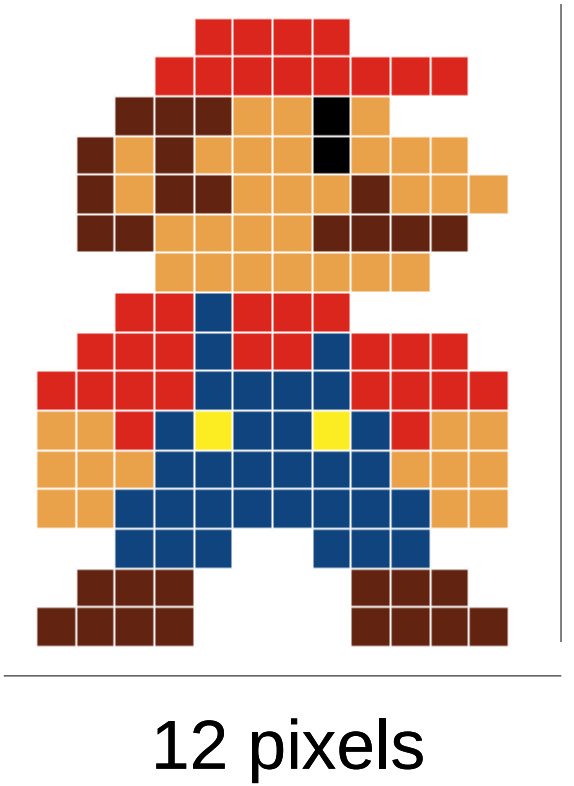
Images

- An image is a matrix.
  - Each cell has an intensity value.
  - Intensity between 0 and 255.
    - 0 for white
    - 255 for black
    - 254 shades of gray in between



Color

- 3 numbers for each pixel:
  - Red intensity (0, 255).
  - Blue intensity (0, 255).
  - Green intensity (0, 255).



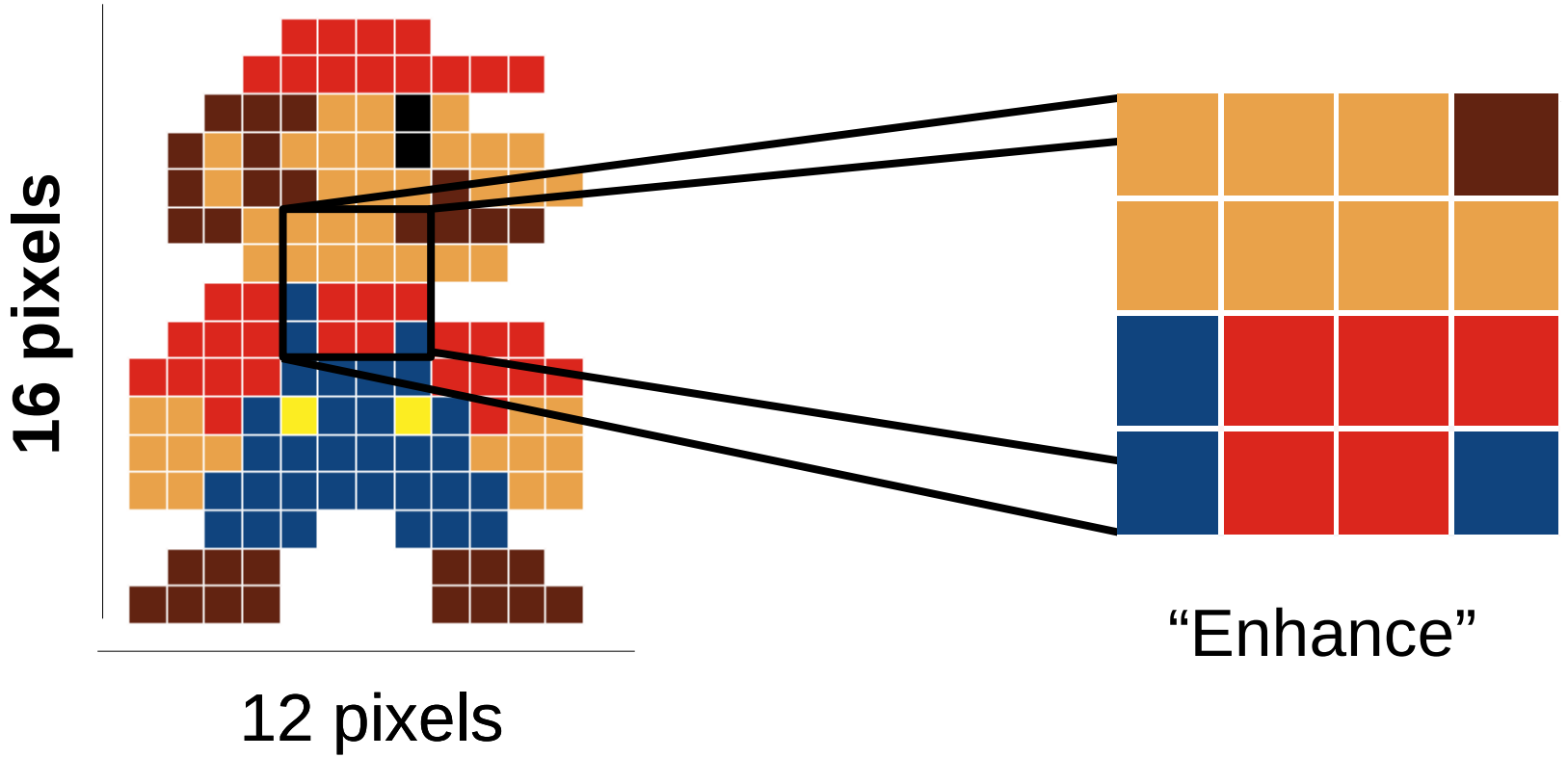
Video

- Sequence of images.

Video

- Sequence of images.
  - Store differences between each pair of frames.

Images



Video

Thanks!