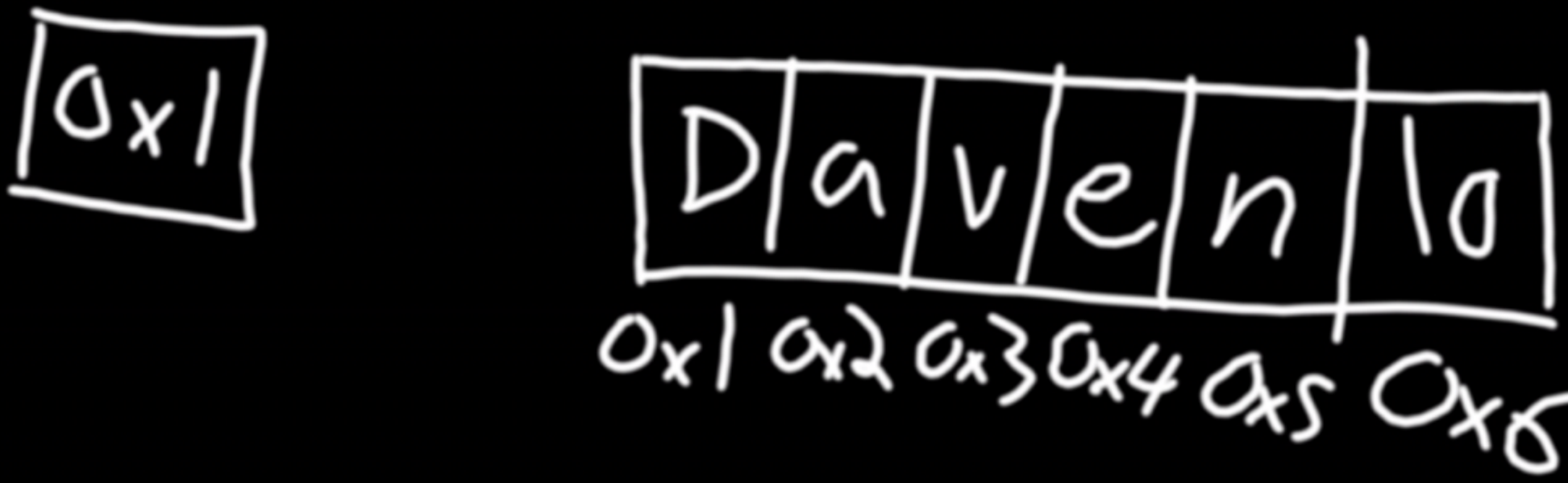
week 5

notes

cs50.harvard.edu/lectures

string

string s = GetString();



char*

```
string s = GetString();
string t = GetString();

if (s == t)
{
    printf("You typed the same thing!\n");
}
else
{
    printf("You typed different things!\n");
}
```

```c
char* s = GetString();
char* t = GetString();

if (s != NULL && t != NULL)
{
    if (strcmp(s, t) == 0)
    {
        printf("You typed the same thing!\n");
    }
    else
    {
        printf("You typed different things!\n");
    }
}
```

```
string s = GetString();
...
string t = s;
if (strlen(t) > 0)
{
    t[0] = toupper(t[0]);
}
```

```
char* s = GetString();
...
char* t = malloc((strlen(s) + 1) * sizeof(char));
...
for (int i = 0, n = strlen(s); i <= n; i++)
{
    t[i] = s[i];
}
...
if (strlen(t) > 0)
{
    t[0] = toupper(t[0]);
}
```

```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

```
void swap(int* a, int* b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

malloc

CS50 Library

```
GetChar

GetDouble

GetFloat

GetInt

GetLongLong

GetString
```

```
int main(void)
{
    int* x;
    int* y;

    x = malloc(sizeof(int));

    *x = 42;

    *y = 13;

    y = x;

    *y = 13;
}
}
```

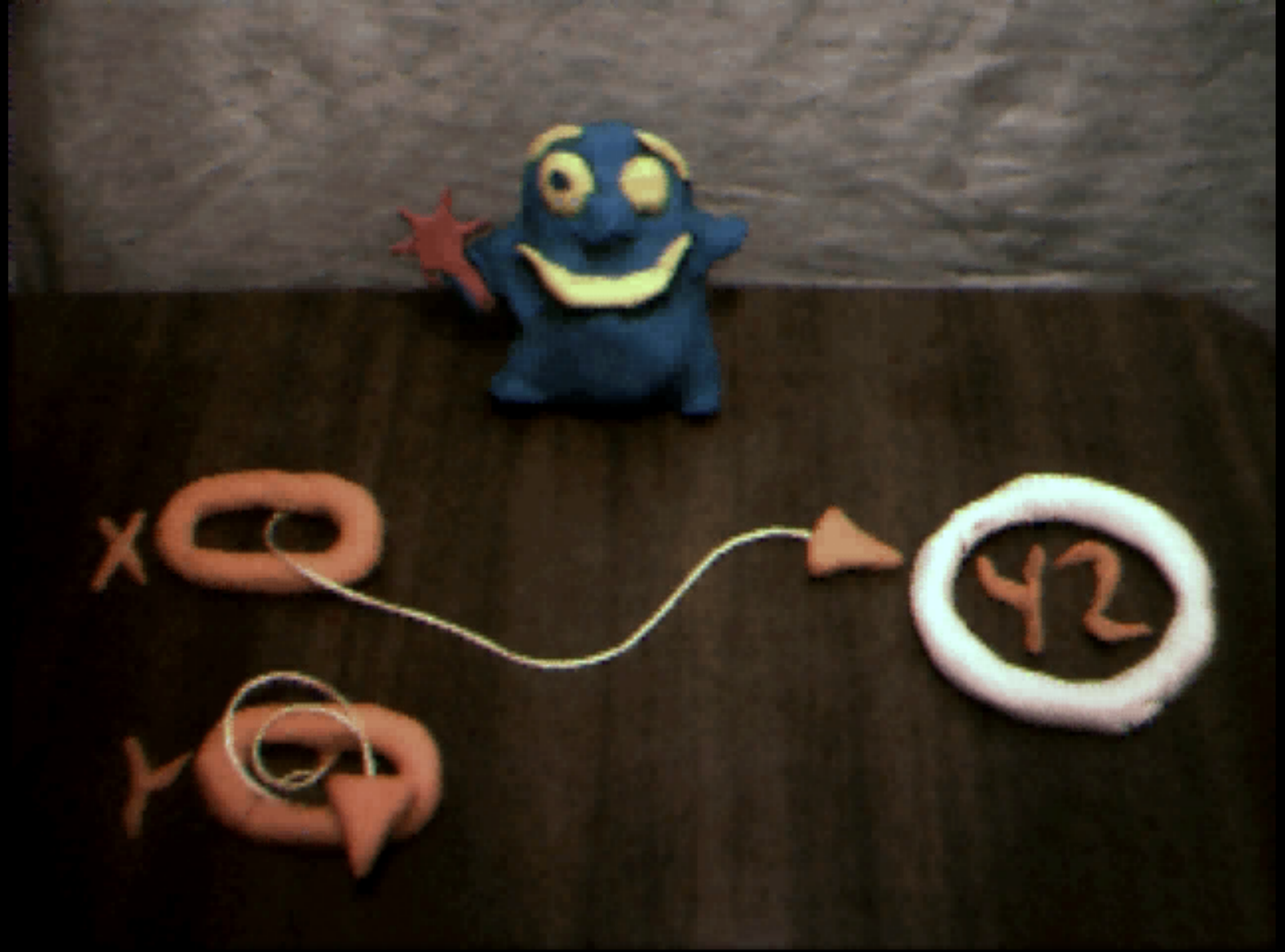
```
int* x;
int* y;
```


```
int* x;
int* y;
```

```
x = malloc(sizeof(int));
```

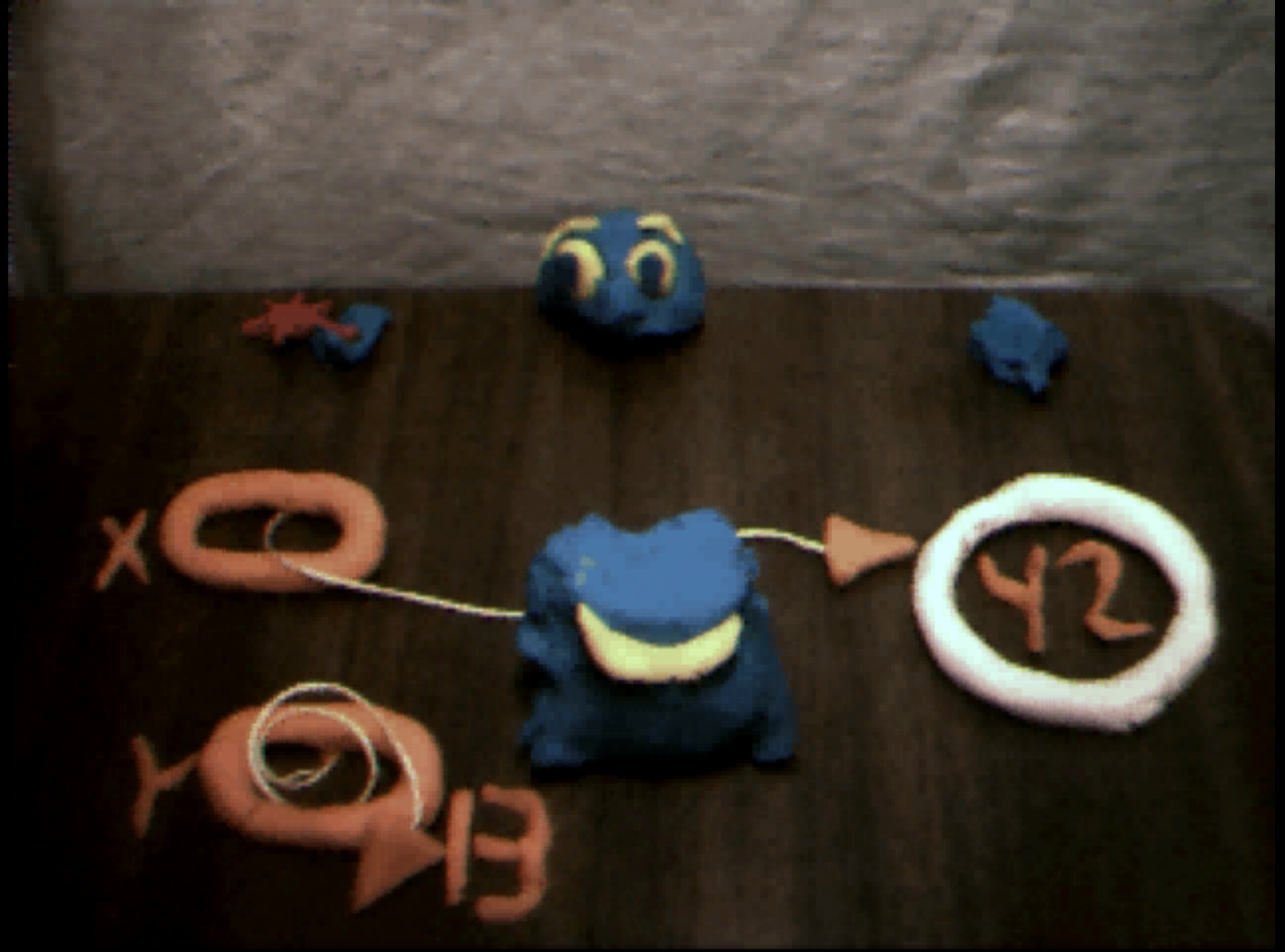

```
x = malloc(sizeof(int));
```

```
*x = 42;
```



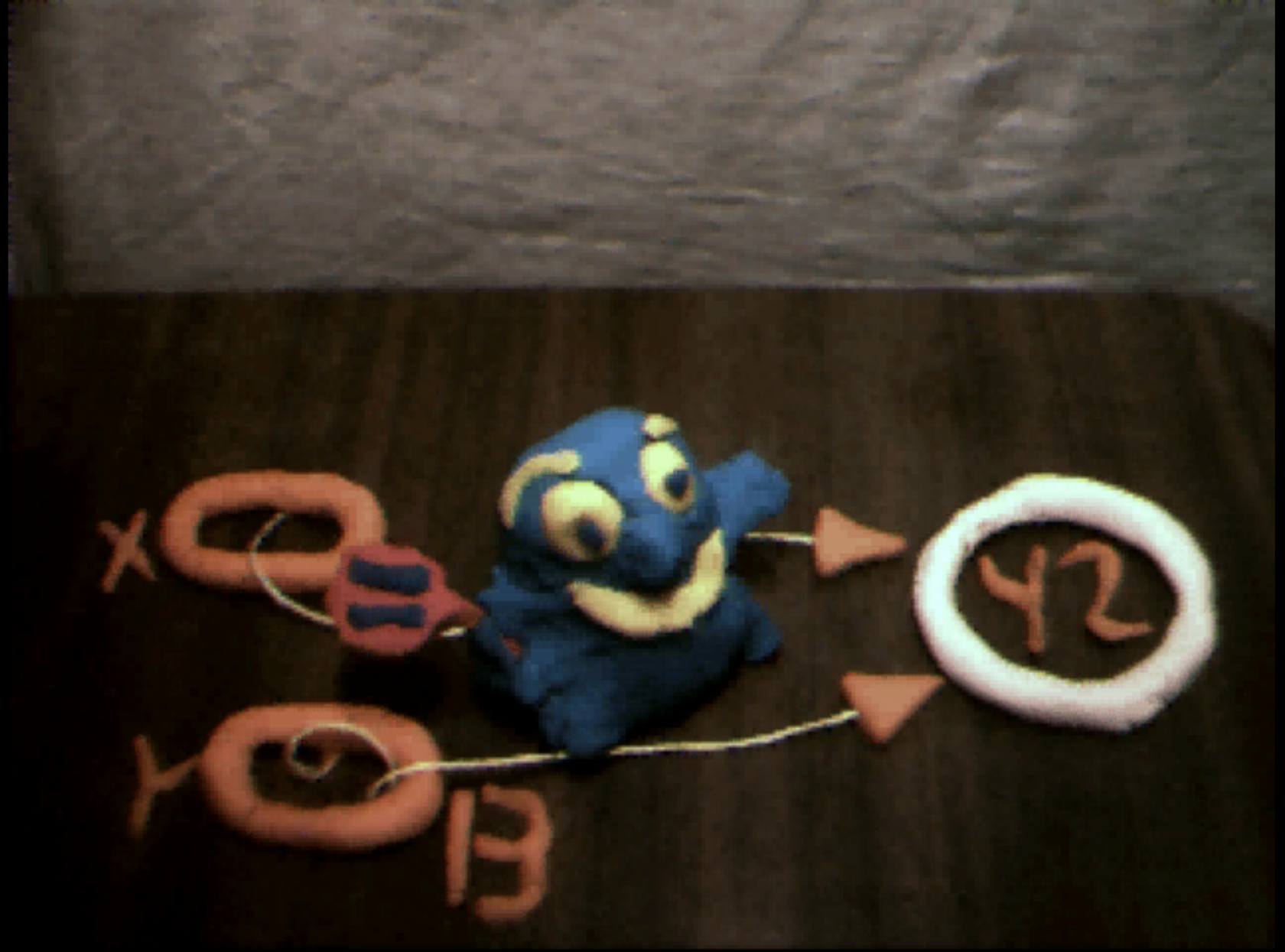

```
*x = 42;
```

```
*y = 13;
```



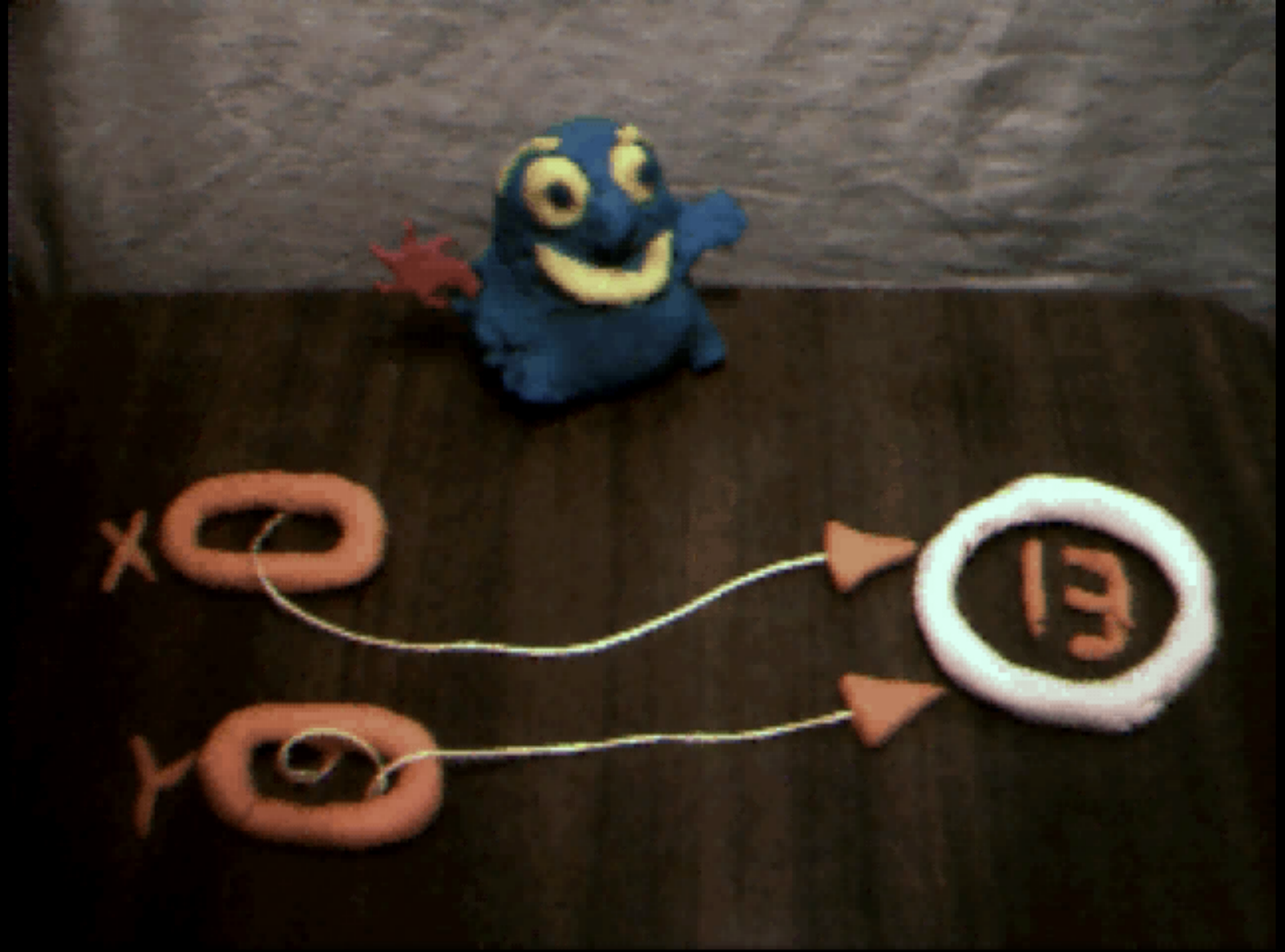

```
*y = 13;
```

```
y = x;
```




```
y = x;
```

```
*y = 13;
```




```
*y = 13;
```

valgrind

```
valgrind --leak-check=full ./program
```

```
Invalid write of size 4
   at 0x804840F: f (memory.c:21)
   by 0x8048421: main (memory.c:26)

40 bytes in 1 blocks are definitely lost in loss record 1 of 1
   at 0x4025BDC: malloc (vg_replace_malloc.c:195)
   by 0x8048405: f (memory.c:20)
   by 0x8048421: main (memory.c:26)
```

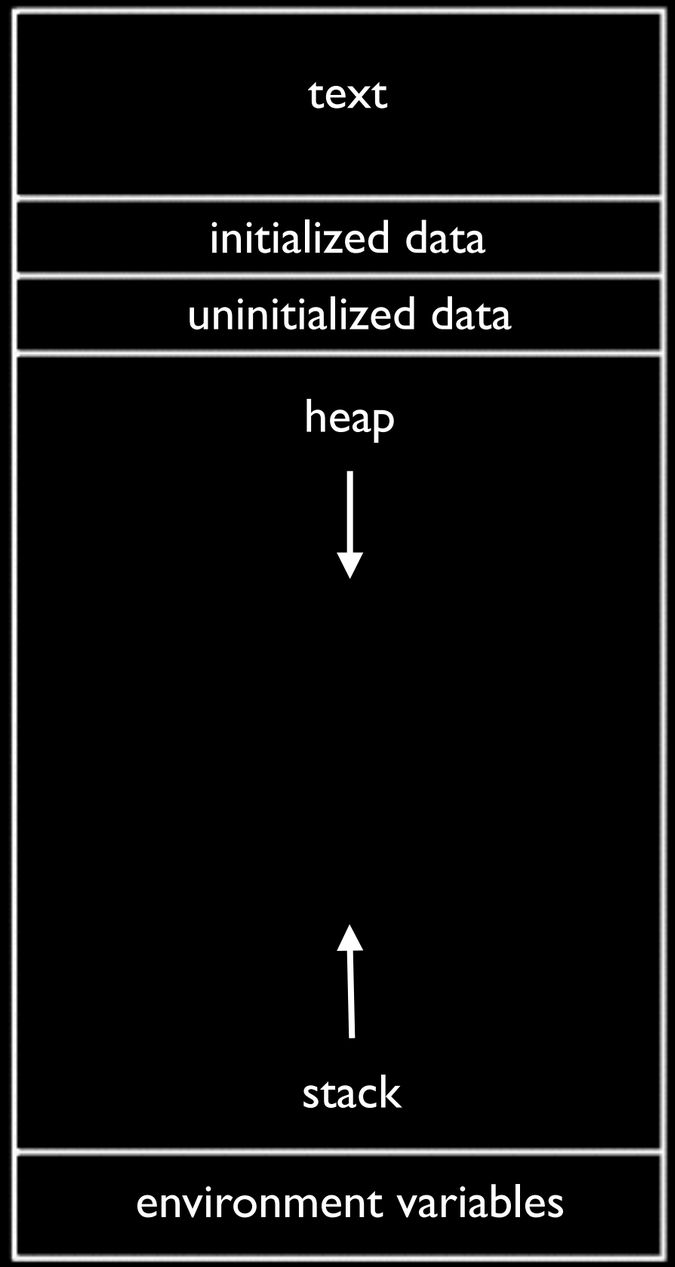
buffer overflow

```
#include <string.h>

void f(char* bar)
{
    char c[12];
    strncpy(c, bar, strlen(bar));
}

int main(int argc, char* argv[])
{
    f(argv[1]);
}
```

adapted from http://en.wikipedia.org/wiki/Stack_buffer_overflow


text
initialized data
uninitialized data
heap
stack
environment variables

http://xkcd.com/138/

to be continued...