hello, section!

week 3

warmup

pset 2 recap

Common Errors

1. Make sure return proper values
2. Computing strlen in for loop
3. Using characters vs. numbers

```c
int index = 0;
for (int i = 0, n = strlen(plaintext); i < n; i++)
{
    if (isalpha(plaintext[i]))
    {
        int key = toupper(keyword[index]) - 'A';
        if (isupper(plaintext[i]))
        {
            printf("%c", ((plaintext[i] - 'A' + key) % 26) + 'A');
        }
        else
        {
            printf("%c", ((plaintext[i] - 'a' + key) % 26) + 'a');
        }
        index = (index + 1) % strlen(keyword);
    }
    else
    {
        printf("%c", plaintext[i]);
    }
}
```

problem solving strategies

problem solving strategies

1. understand the problem
2. devising a plan
3. carrying out the plan
4. test your code

1. understand the problem

do you understand the spec?
could you explain it to a friend?

2. devise a plan

what steps will you take?
can you outline your solution in pseudocode?

3. carry out the plan

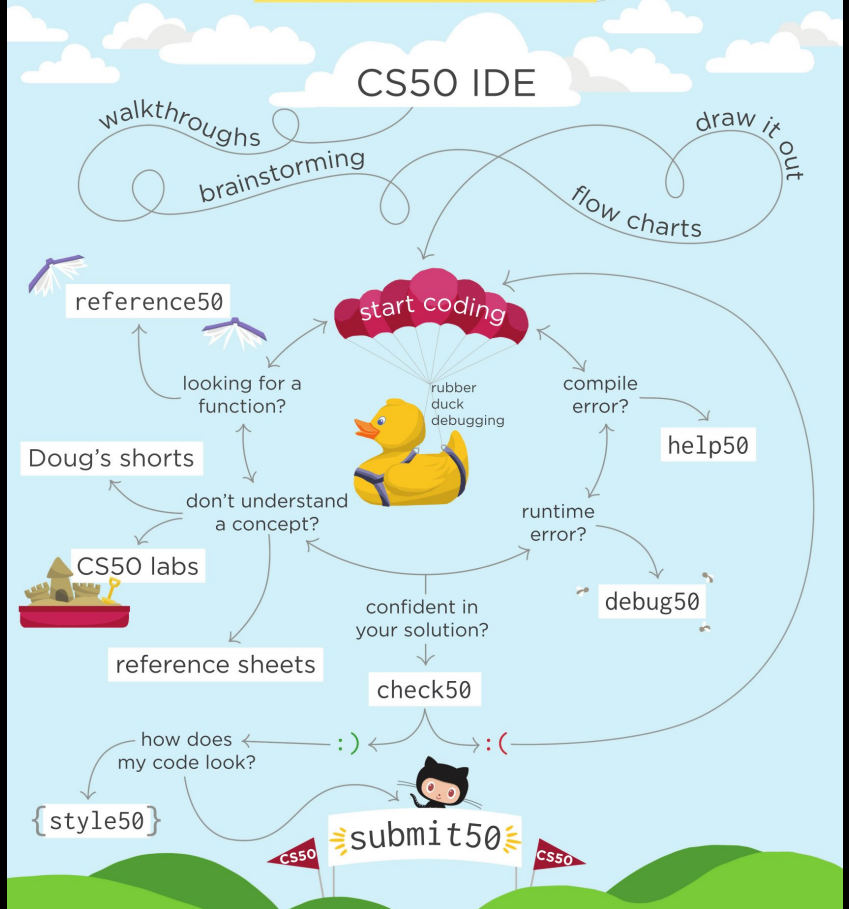
looking at code examples, how can you implement your ideas?
use reference50 & google for help!

4. test your code, check your work

check50, debug50, CS50 discourse, OHs
what exactly are your bugs?
where can you improve?

A GUIDE TO CS50 Tools

start programming here!



file I/O

file I/O

input

- user prompt
- command line argument

output

- print to console
- return

file I/O

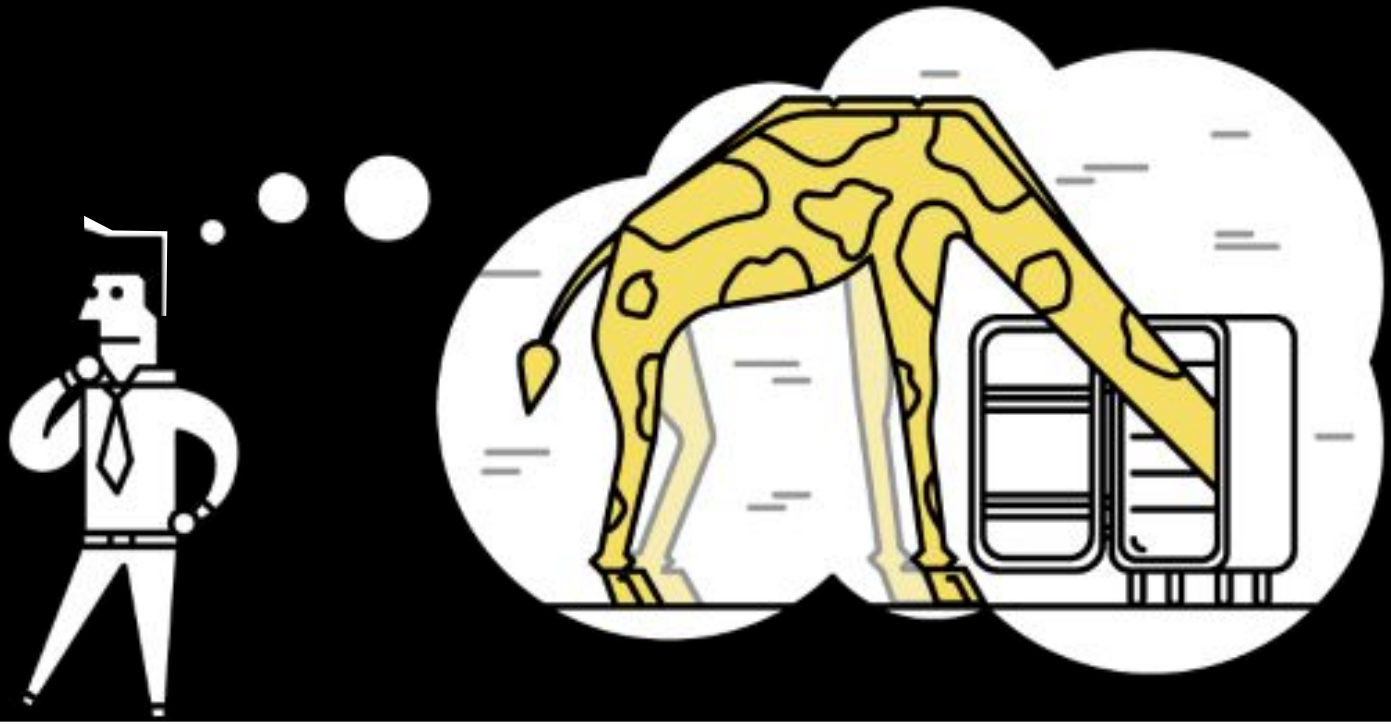
input

- user prompt
- command line argument
- read from file

output

- print to console
- return
- write to file

file I/O



file I/O

1. open your input file (read)
2. open your output file (write)
3. read from your input file
4. process data
5. write to your output file
6. close all files (free memory)

file I/O

```
FILE *in = fopen("input.txt", "r");
```

file I/O

```
FILE *out = fopen("output.txt", "w");
```

file I/O

```
int c = fgetc(in);
fputc(c, out);
```

file I/O

```
fclose(in);
fclose(out);
```

file I/O - stdio library

`fopen()` -- creates a file reference

`fread()` -- reads some amount of data from a file

`fwrite()` -- writes some amount of data to a file

`fgets()` -- reads a single string from a file (typically, a line)

`fputs()` -- writes a single string to a file (typically, a line)

`fgetc()` -- reads a single character from a file

`fputc()` -- writes a single character from a file

`fseek()` -- like rewind and fast forward on YouTube, to navigate around a file

`ftell()` -- like the timer on YouTube, tells you where you are in a file (how many bytes in)

`fclose()` -- closes a file reference, used once done working with the file

pointers

literally pass values between functions

pointers

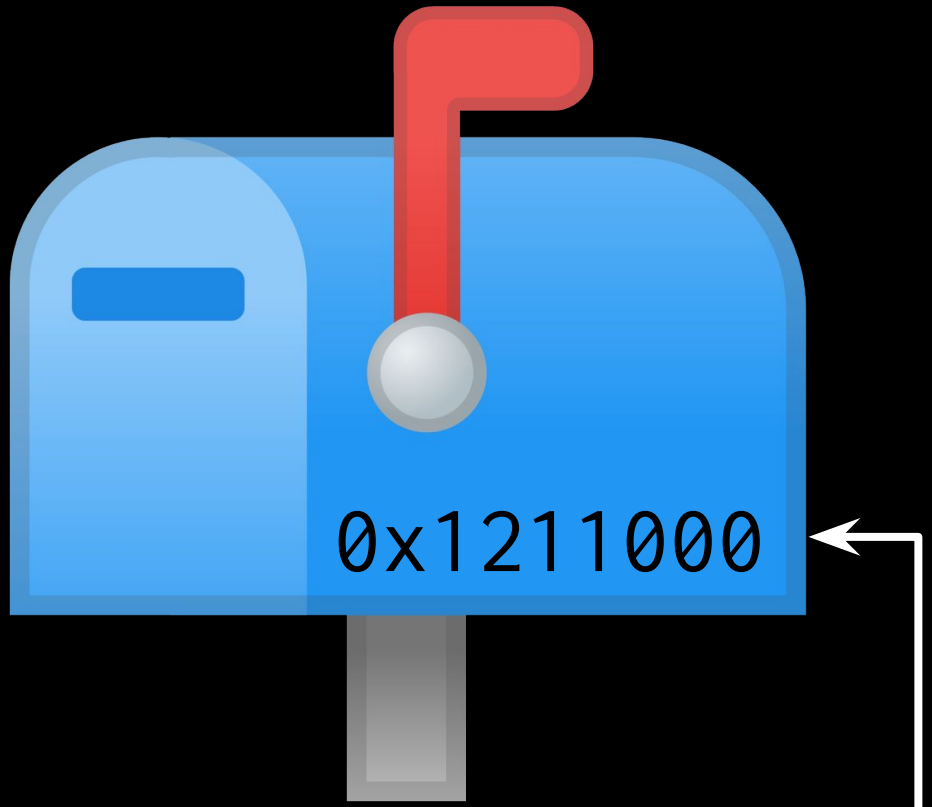
a pointer is just an address

a pointer is just an address

a pointer is just an address

a pointer is just an address

pointers



pointers

pointers

pointers

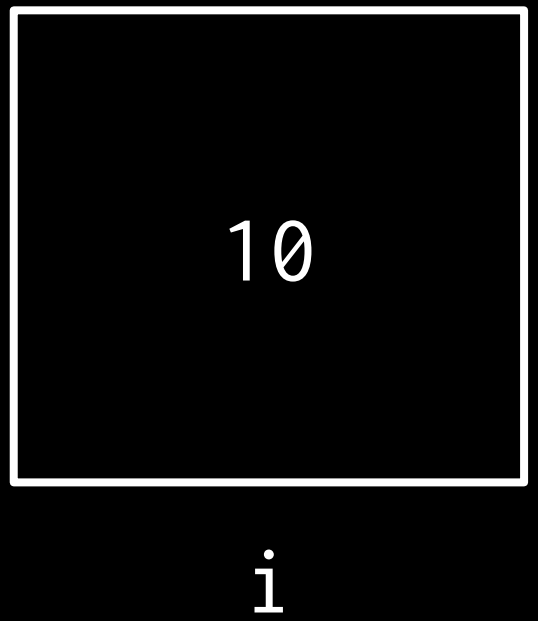
a pointer's...

value → memory address

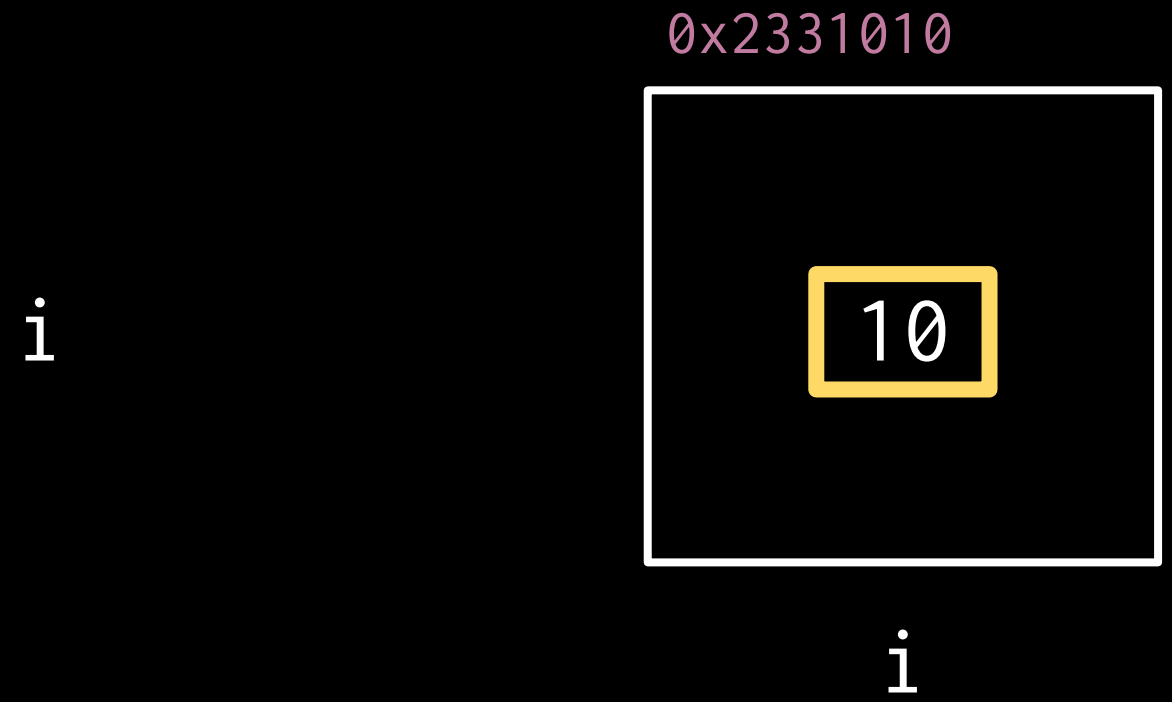
type → type of data @ memory address

pointers

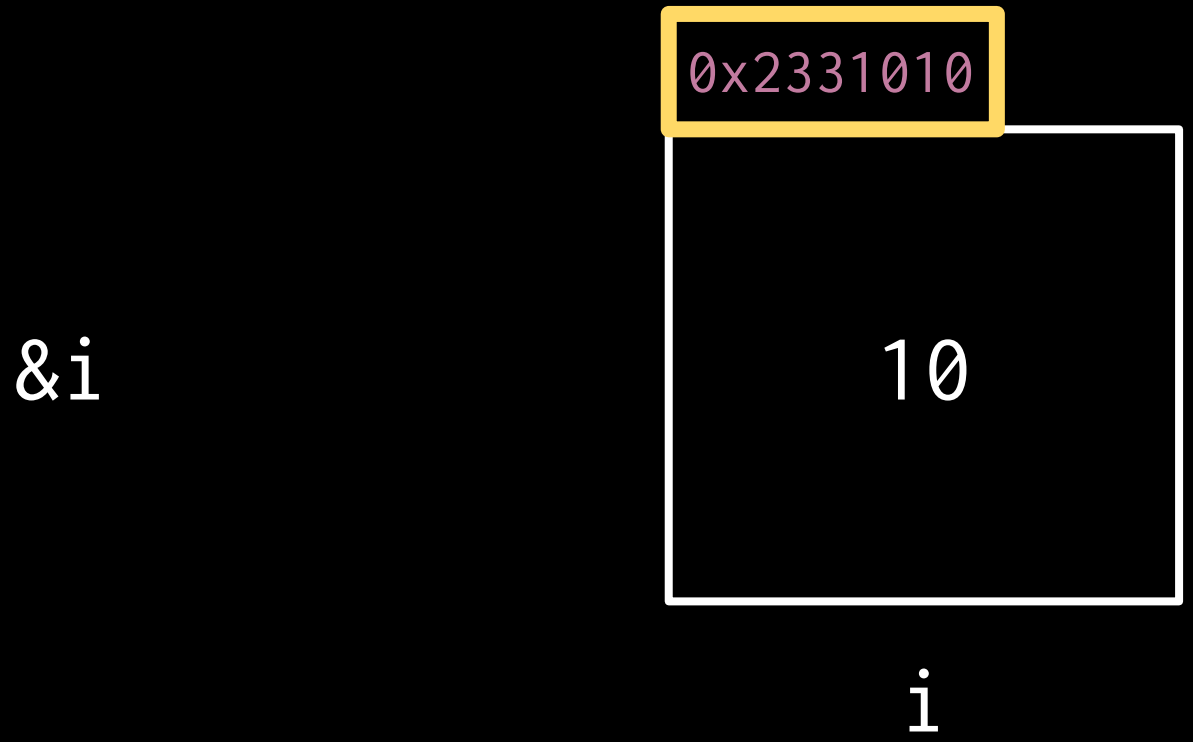
```
int i = 10;
```



pointers

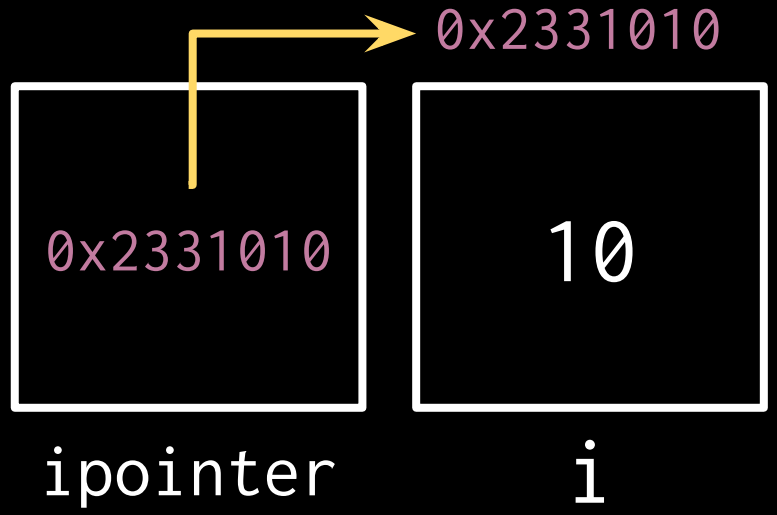


pointers



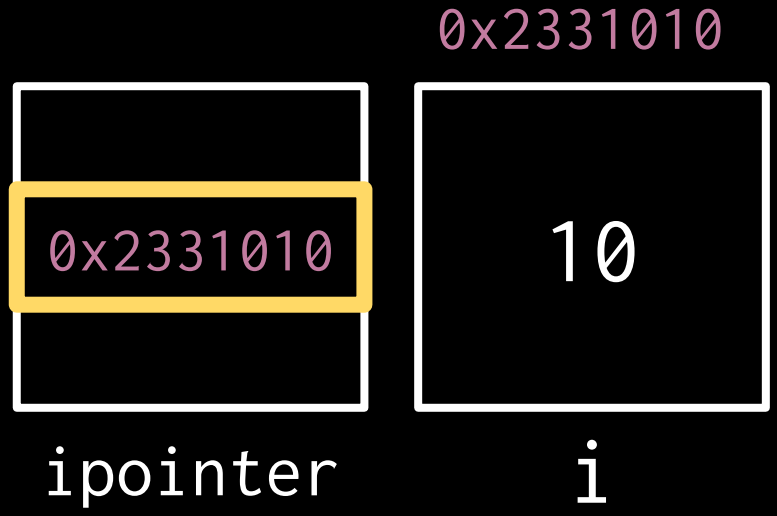
pointers

```
int* ipointer = &i;
```



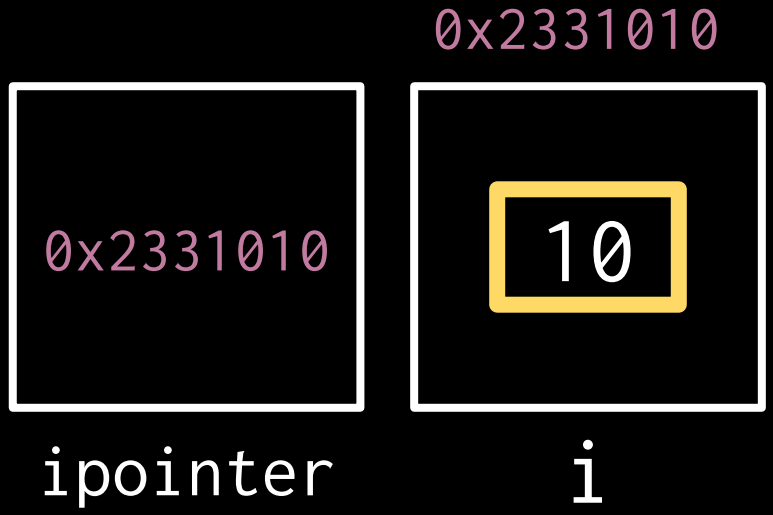
pointers

ipointer



pointers

*ipointer



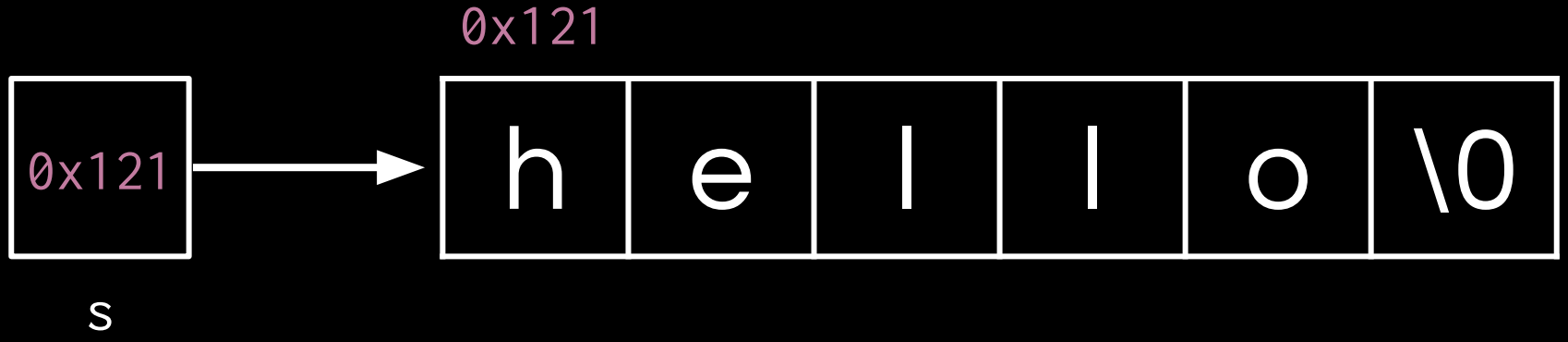
pointers

char*

~~strings~~

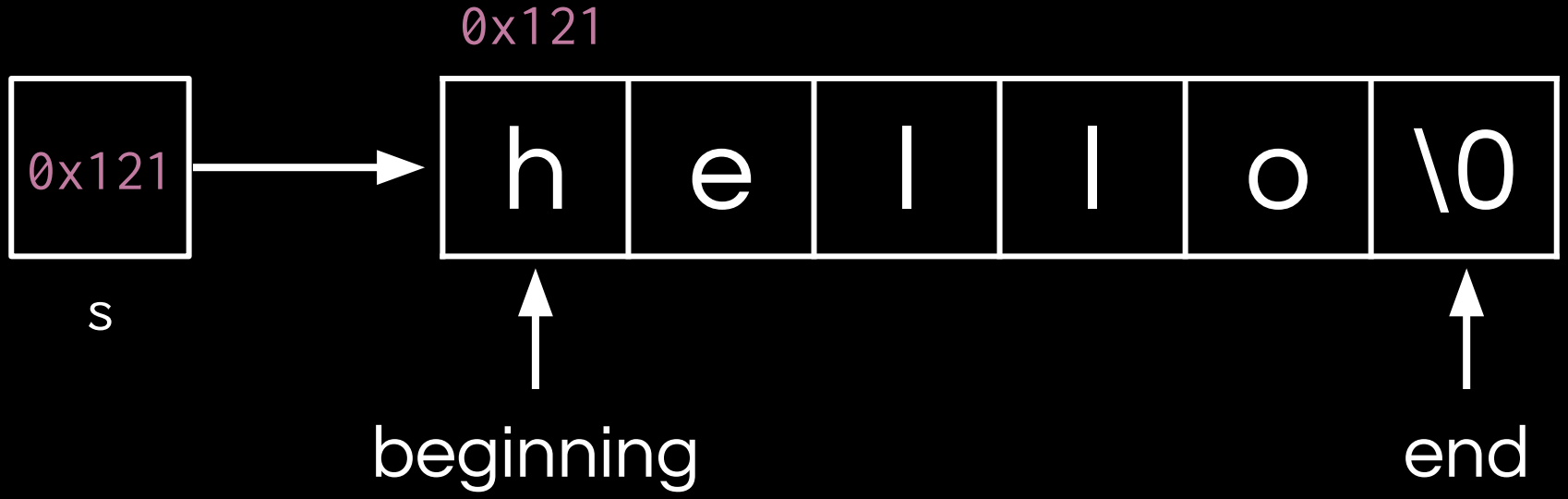
pointers

```
char* s = “hello”
```

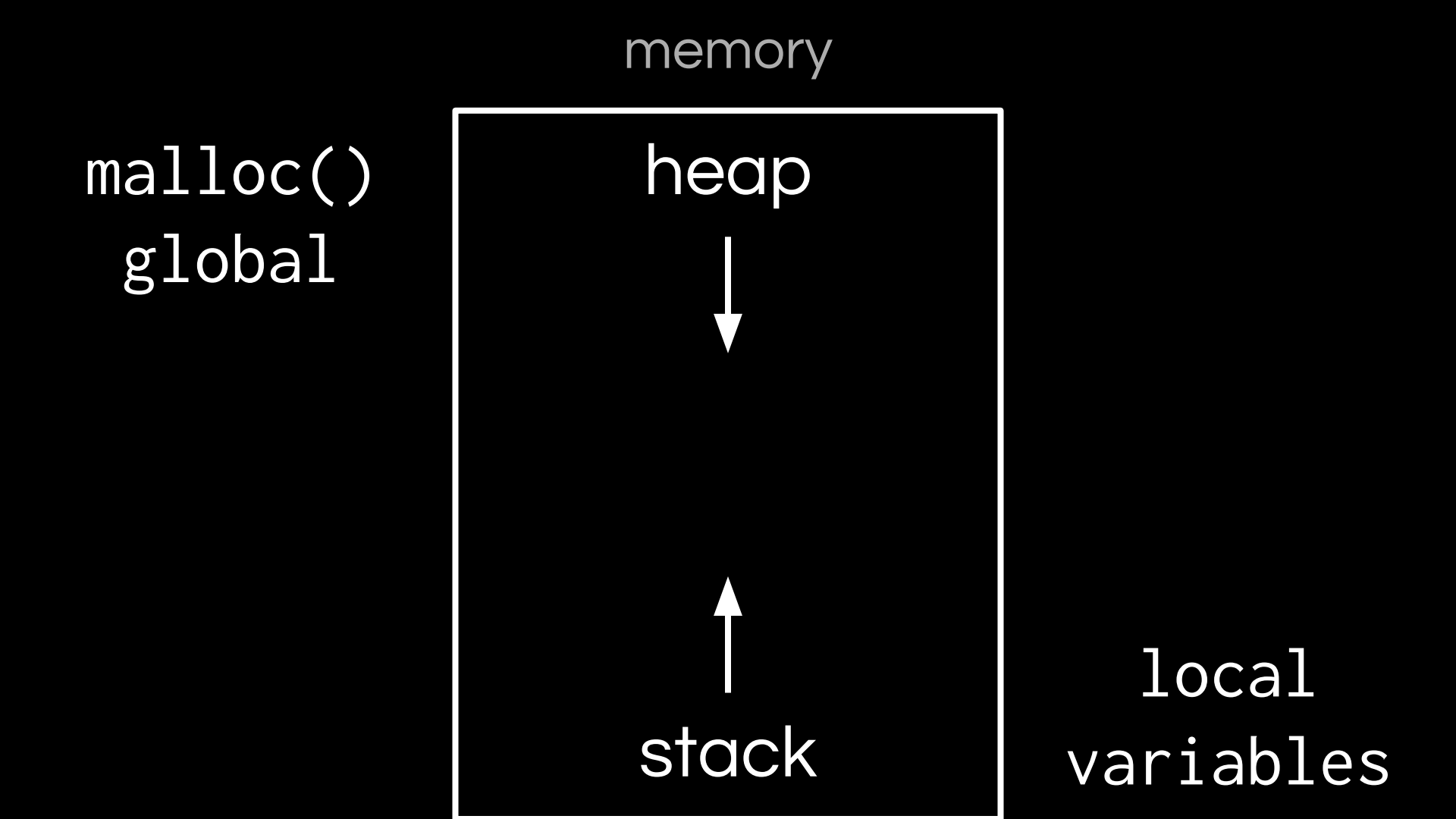


pointers

```
char* s = “hello”
```

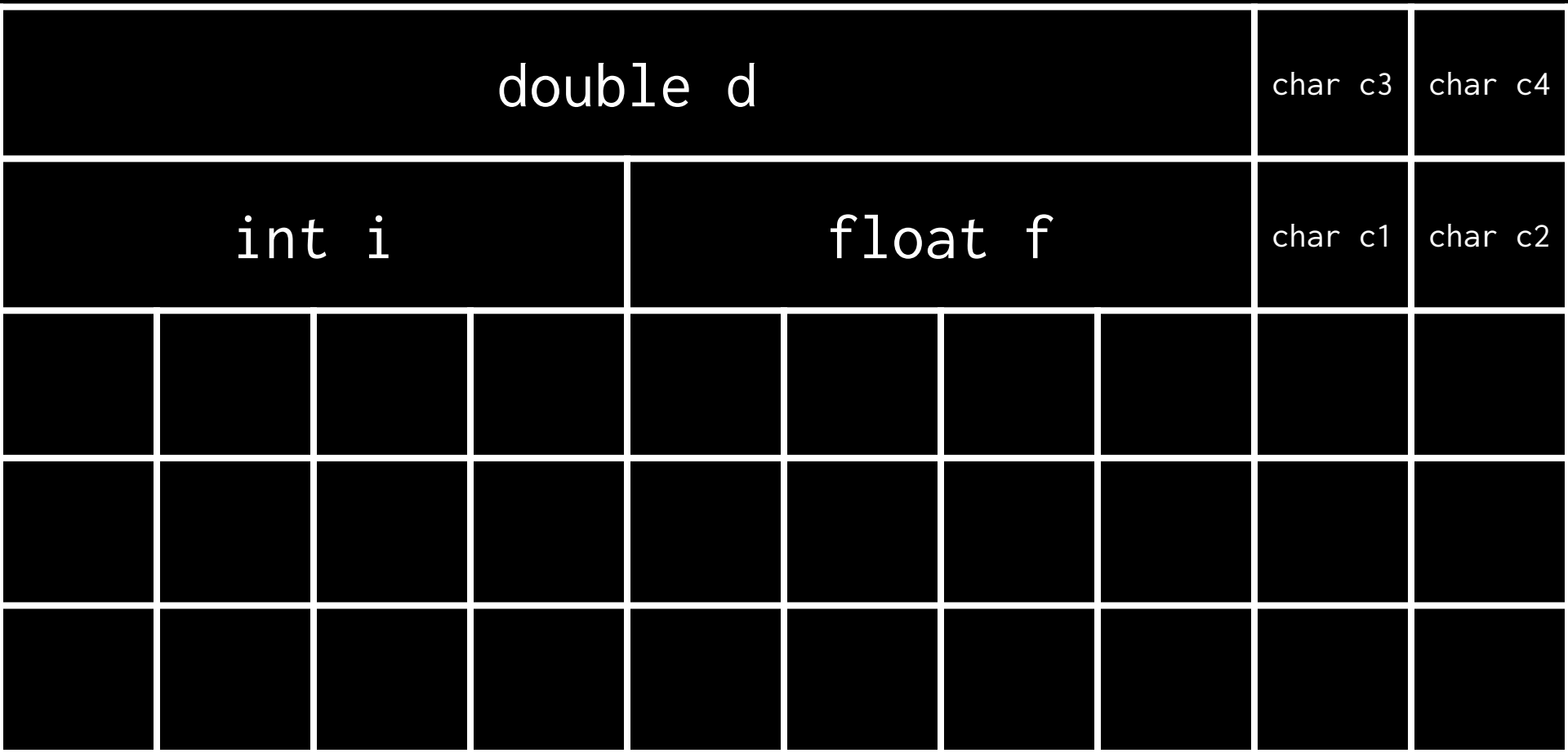


memory

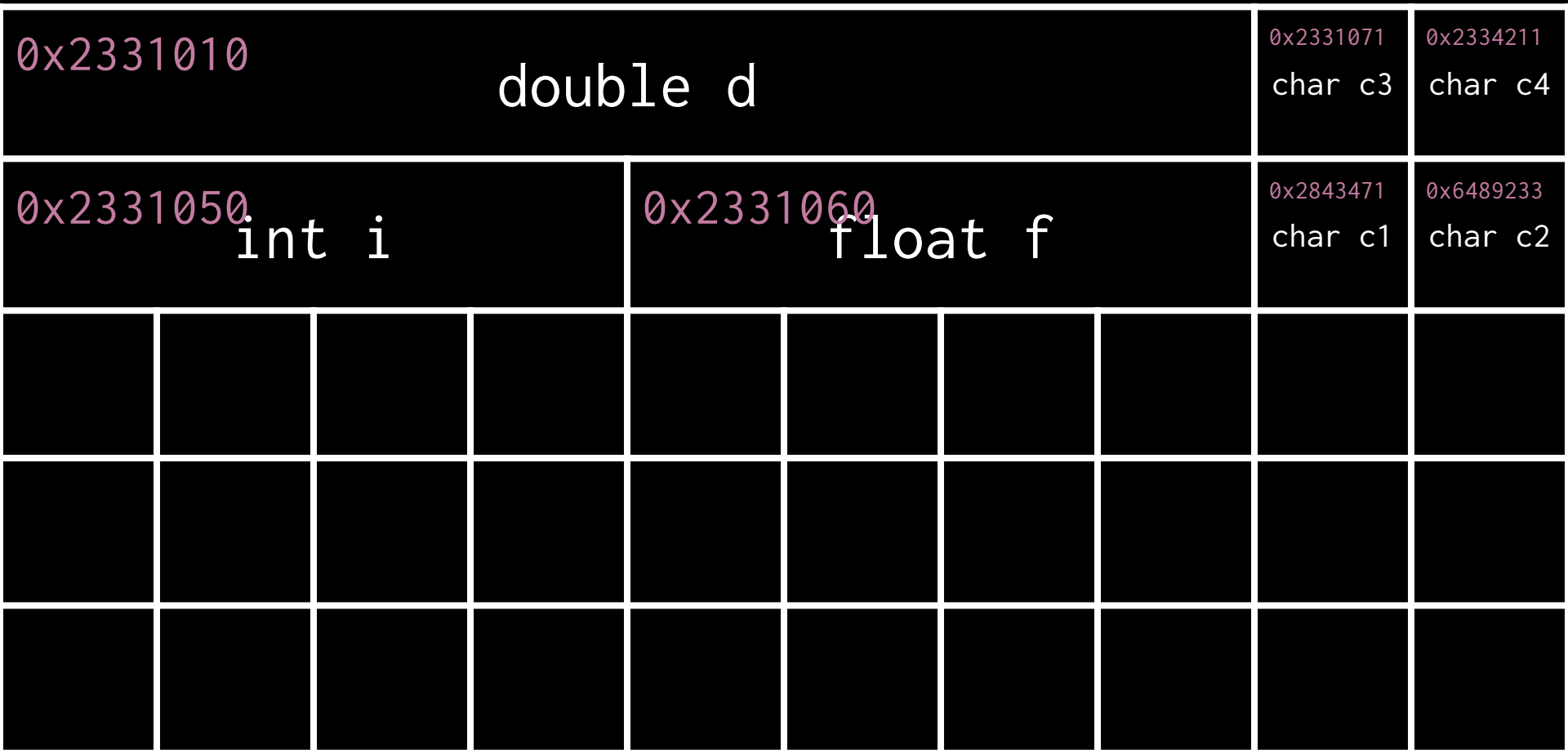

memory
malloc()
global
heap
stack
local
variables

pointers

pointers



pointers


0x2331010
double d
0x2331071
char c3
0x2334211
char c4
0x2331050
int i
0x2331060
float f
0x2843471
char c1
0x6489233
char c2

pset requirements

- read from and write to files
- memory management (use `malloc()`)
  - use pointers!
- hexadecimal
- use the tools available to you!

be sure to use CS50 IDE!