Pointers

Help Sessions

Check timetable!

Research Project Signup Reminder

- Help shape the future of DCC
- Volunteer 30 mins
- Entirely optional
- No bearing on your course

Revision sessions Week 6

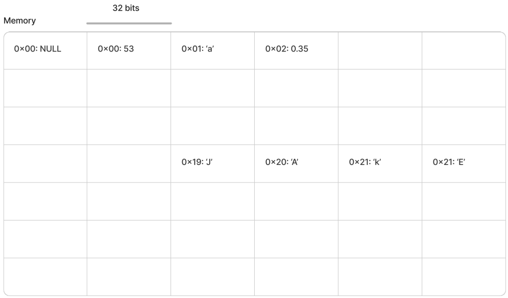
- Tuesday 2pm-4pm (Online)
- Wednesday 3pm-5pm (K17 Seminar Room)

Pointers

Pointers

- All data (variables) are stored in **memory**
- You can think of memory as a big grid
- Each segment of this grid has a unique identifier

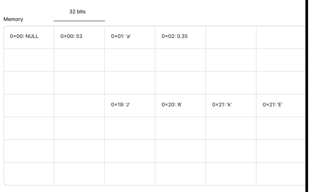
Visualising memory with addresses



0×00: NULL	0×00: 53	0×01: 'a'	0×02: 0.35		
		0×19: 'J'	0×20: 'A'	0×21: 'k'	0×21: 'E'

So far, we have only dealt with values

- We can also access the address
- By storing that address in a variable, we have a **pointer**



0×00: NULL	0×00: 53	0×01: 'a'	0×02: 0.35		
		0×19: 'J'	0×20: 'A'	0×21: 'k'	0×21: 'E'

Pointer Syntax

To declare a pointer

```
<type> * <name_of_variable>
```

^ The `*` means don't request the storage to store `<type>`, but requests memory to store a memory address of `<type>`

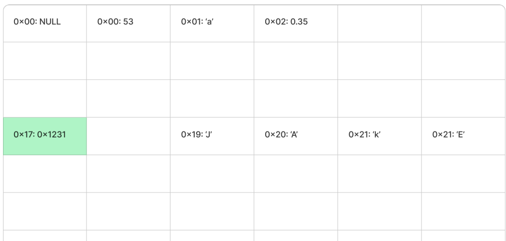
Syntax example:

`int *pointer`

`struct student *student`

Visualise pointer declaration

```c
// declare a pointer to an integer
int *number; // operating system returns 0x17
```



0×00: NULL	0×00: 53	0×01: ‘a’	0×02: 0.35		
🟩 0×17: 0×1231		0×19: ‘J’	0×20: ‘A’	0×21: ‘k’	0×21: ‘E’

Address of operator `&`

- What if we want to query what the address of a variable is?
- We can use the address_of operator:

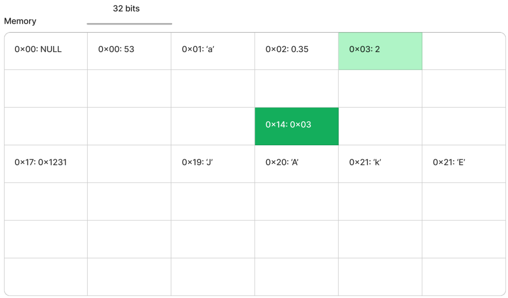
`&`

Syntax of address of: `&<variable>`

Example

```
int number = 2;
&number // the address
of number
```

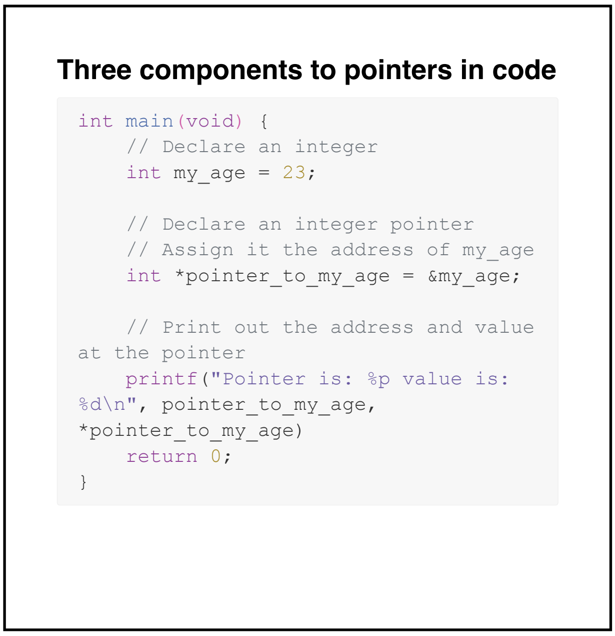
```
int number = 2;


int *pointer_to_number =
&number
```



0×00: NULL	0×00: 53	0×01: 'a'	0×02: 0.35	0×03: 2	
			0×14: 0×03		
0×17: 0×1231		0×19: 'J'	0×20: 'A'	0×21: 'k'	0×21: 'E'

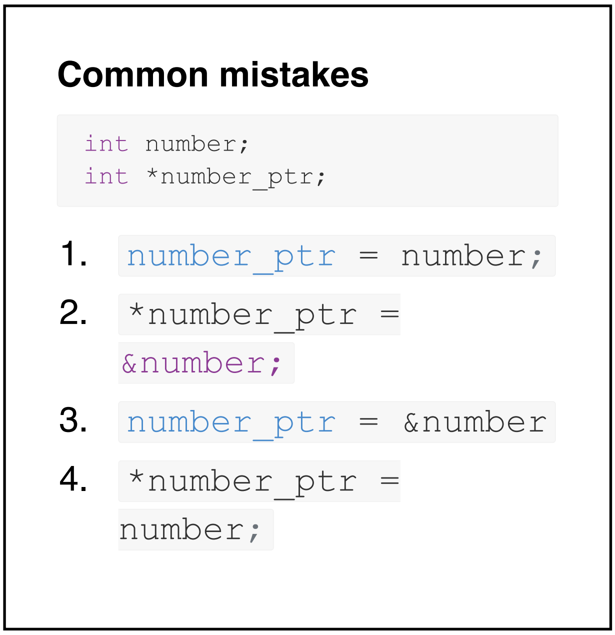
Dereferencing

- Dereferencing is simply accessing the value at the address of a pointer
- It uses the `*` symbol again (which causes confusion)
- `*my_int_pointer` -> will get the integer at the address location



Three components to pointers in code

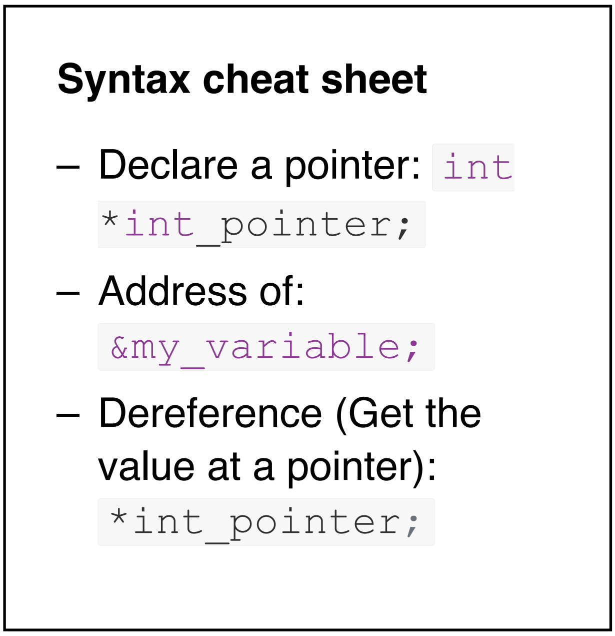
```
int main(void) {
    // Declare an integer
    int my_age = 23;

    // Declare an integer pointer
    // Assign it the address of my_age
    int *pointer_to_my_age = &my_age;

    // Print out the address and value
at the pointer
    printf("Pointer is: %p value is:
%d\n", pointer_to_my_age,
*pointer_to_my_age)
    return 0;
}
```



Common mistakes

```
int number;
int *number_ptr;
```

1. `number_ptr = number;`
2. `*number_ptr = &number;`
3. `number_ptr = &number`
4. `*number_ptr = number;`



Syntax cheat sheet

- Declare a pointer: `int *int_pointer;`
- Address of: `&my_variable;`
- Dereference (Get the value at a pointer): `*int_pointer;`

Demo

But JAKE, why are they *USEFUL*

- Let's look at an example with arrays and parameters

Feedback

https://forms.office.com/r/Ze4admEWnR