Pointers, etc.

CS 211

Road map

A program to sort numbers (using an array)

Road map

A program to sort numbers (using an array)

C variables don’t share their objects

Road map

A program to sort numbers (using an array)

C variables don’t share their objects

A pointer is an object that refers to another object

Initial code setup

The code in this course is available in your Unix shell account. You can get your own copy like this:

```
% cd cs211
% tar -xvkf ~cs211/lec/04_pointers.tgz
⋮
% cd 04_pointers
```

Up next

A program to sort numbers (using an array)

C variables don’t share their objects

A pointer is an object that refers to another object

Up next

A program to sort numbers (using an array)

C variables don’t share their objects

A pointer is an object that refers to another object

Up next

A program to sort numbers (using an array)

C variables don’t share their objects

A pointer is an object that refers to another object

Understanding * and * and &

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

Understanding * and * and &

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*[1]	`&o`

[1] & doesn't appear in types in C, but it will in C++

Understanding * and * and &

operator	in types	in expressions
*	T*	*p
&	*N/A*	&o

- T can be any type

Understanding `*` and `*` and `&`

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

- `T` can be any type (like `int` or `struct dog` or `struct dog*`)

Understanding * and * and &

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

- `T` can be any type (like `int` or `struct dog` or `struct dog*`)
- `p` can be any expression whose value is a pointer

Understanding * and * and &

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

- `T` can be any type (like `int` or `struct dog` or `struct dog*`)
- `p` can be any expression whose value is a pointer
- `o` can be any object

Understanding `*` and `*` and `&`

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

- `T` can be any type (like `int` or `struct dog` or `struct dog*`)
- `p` can be any expression whose value is a pointer
- `o` can be any object (like `x` or `a[3]`, but not a non-object value like `5` or `n + m`)

Understanding * and * and &

operator	in types	in expressions
`*`	`T*`	`*p`
`&`	*N/A*	`&o`

- `T` can be any type (like `int` or `struct dog` or `struct dog*`)
- `p` can be any expression whose value is a pointer
- `o` can be any object (like `x` or `a[3]`, but not a non-object value like `5` or `n + m`)

Then `*p` is the object that `p` points to, and `&o` is a pointer to the object `o`.

– Next time: Dynamic Memory –