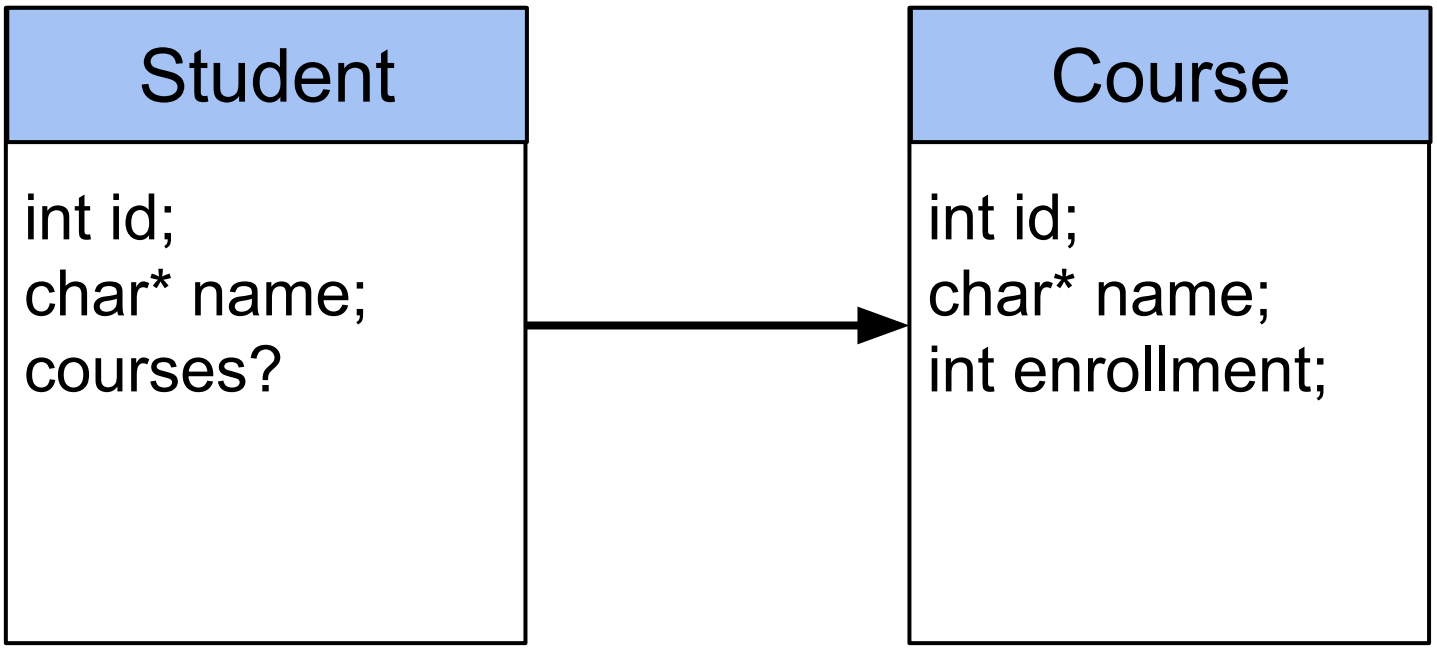
structs

Adapted from materials by Dr. Carrier



Object-oriented programming in C

Object-oriented programming in C (kinda)

Object-oriented programming in C (kinda)

- We can *structure* our data
    - We're not limited to raw ints, chars, arrays, etc

structs

Structs allow us to group together pieces of data

structs

Structs allow us to group together pieces of data

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

structs

Structs allow us to group together pieces of data

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

We can treat `Coord` like a new type!

structs

Structs allow us to group together pieces of data

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

We can treat `Coord` like a new type!

To create and use an *instance* of a struct:

structs

Structs allow us to group together pieces of data

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

We can treat `Coord` like a new type!

To create and use an *instance* of a struct:

```
Coord c;
c.x = 5.0;
c.y = 10;
```

structs

- Structs can container raw data types, pointers, arrays, even other structs!

structs

- Structs can container raw data types, pointers, arrays, even other structs!
- We can access members of a struct with `.`
  - E.g., `c.x = 5;` `double d = c.y;`

structs + pointers

- Structs still work with pointers!

structs + pointers

- Structs still work with pointers!
- We can create a pointer to a struct:

```
Coord* p = &c;
```

structs + pointers

- Structs still work with pointers!
- We can create a pointer to a struct:

```
Coord* p = &c;
```

- We can access members with ->

```
p->x = 1.0;
```

structs + pointers

- Structs still work with pointers!
- We can create a pointer to a struct:

```
Coord* p = &c;
```

- We can access members with ->

```
p->x = 1.0;
```

  - This is the same as `(*p).x = 1.0;`

structs + pointers

- Structs still work with pointers!
- We can create a pointer to a struct:

```
Coord* p = &c;
```

- We can access members with ->

```
p->x = 1.0;
```

  - This is the same as `(*p).x = 1.0;`

We can allocate structs on the stack or the heap!

structs

What are we missing?

structs

What are we missing?

Methods! (member functions)

structs

What are we missing?

Methods! (member functions)

C does not support methods by default.

structs

What are we missing?

Methods! (member functions)

C does not support methods by default.

You can build them, but it’s complicated

(function pointers)

Details

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

```
Coord c;
```

Details

```
typedef struct Coord {
  double x;
  double y;
} Coord;
```

```
Coord c;
```

This is only one way to declare a struct.

```
struct Coord2 {
  double x;
  double y;
};
struct Coord2 example;
```

typedef

typedef

typedef defines a new type!

```
typedef type name;
```

typedef

typedef defines a new type!

```
typedef type name;
```

This is not limited to structs:

```
typedef int* int_pointer;
int x = 40;
int_pointer p = &x;
printf("p: %p; *p: %d\n", p, *p);
```

typedef

typedef defines a new type!

`typedef type name;`

This is not limited to structs:

```
typedef int* int_pointer;
int x = 40;
int_pointer p = &x;
printf("p: %p; *p: %d\n", p, *p);
```

With structs, this prevents us from typing "`struct Coord`" over and over again!