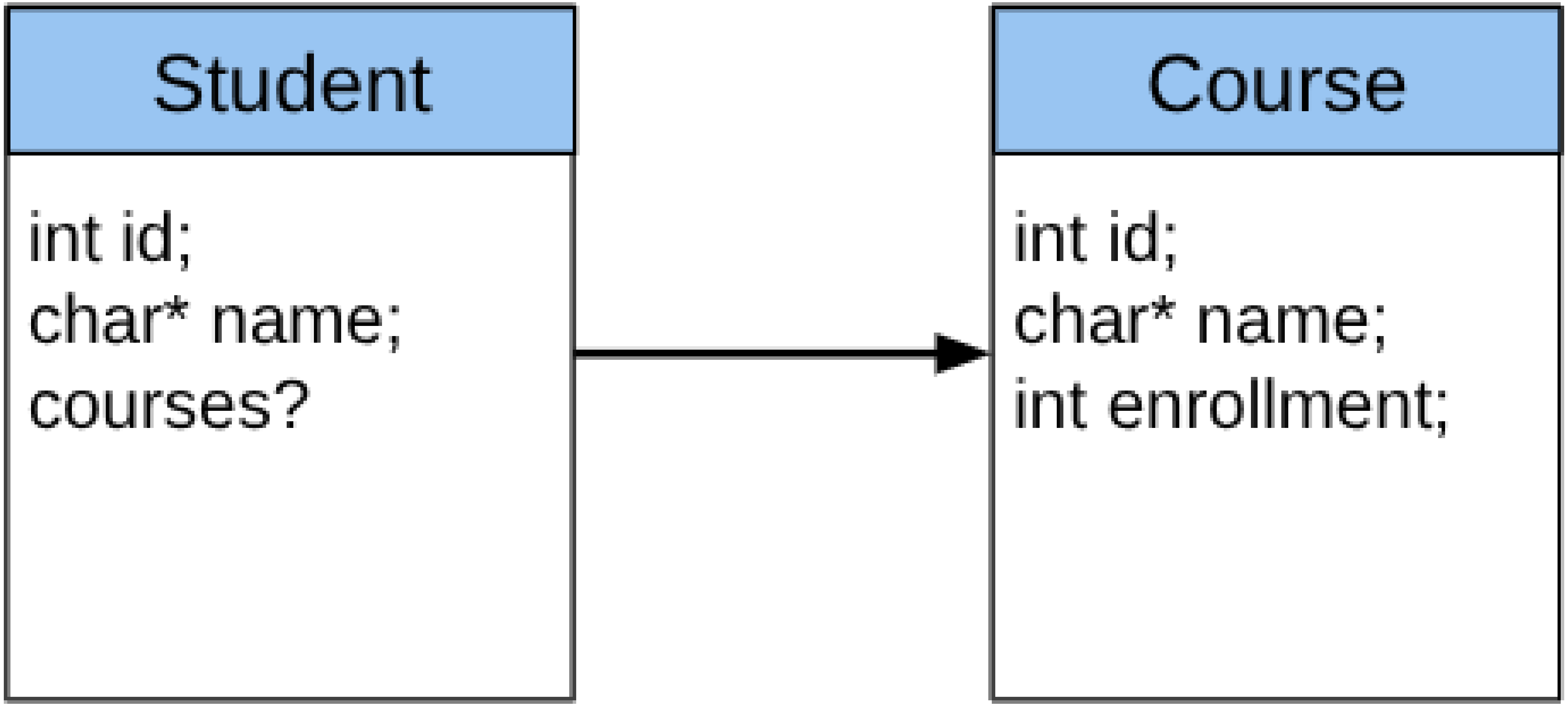
CIS241

System-Level Programming and Utilities

C - structs

Erik Fredericks, frederer@gvsu.edu
Fall 2025

> **Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers**


Student
int id;
char* name;
courses?
Course
int id;
char* name;
int enrollment;

Object-oriented programming in C

(somewhat)

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

```
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!

We can access members of a struct with .

- E.g., `c.x = 5;` `double d = c.y;`

structs

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!

What's missing?

That OO thing...

Methods! (member functions)

C does not support methods by default.

- You can build them, but it’s complicated (function pointers)

More on structs (declaring)

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

Also can do:

```
struct Coord2 {
    double  x;
    double  y;
}
...
struct Coord2 d;
```

typedef

typedef defines a new type!

- `typedef type name;`

This is not limited to structs:

```c
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!

Practice!

1. Create a struct that represents a player (name, health, room ID)
2. Set the defaults in a function
3. Use `l` and `r` as inputs to move them around (`l` moves left and `r` moves right)

Skeleton (curly braces on same line for space)

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef  struct Player {
} Player;

void  setupPlayer(?)  {
}

int  main()  {
     return  0;
}
```

The struct

```
typedef  struct Player {
  int roomID;
  int health;
  char* name;
} Player;
```

The setup function

```c
void  setupPlayer(Player* p)  {
  p->roomID = 0;
  p->health = 10;
  p->name = "Erik";
}
```

More main

```
int main() {
    // create and setup player
    Player player;
    setupPlayer(&player);

    bool done = false;  // forever loop variable
    char ch;               // user input

    // forever loop
    while (!done) {

    }

    printf("Done.\n");
    return 0;
}
```

and the loop

```
// forever loop
while (!done) {
   printf("---\n");
   printf("%d %d %s\n", player.roomID, player.health, player.name);
   printf("Waiting for input [l, r, q]: ");

   // read a character - NOTE THE SPACE
   int res = scanf(" %c", &ch);
   if(res == EOF) done = true; // Ctrl+D pressed
   else {
      if (ch == 'l') player.roomID--;
      else if (ch == 'r') player.roomID++;
      else if (ch == 'q') done = true;
   }
   printf("---\n\n");
}
```