Love and Structures

A Valentine's Day Manifesto for CMSC 212
by Asad Sayeed
Love Your Assignment

- You should be starting on Project 1b
  - Do you have any questions you think I can answer here and now?
  - Make sure that you are in the Linux environment.
- There is a quiz today.
  - Do you have any questions about that?
Love Your Quiz!

Quiz!
Joy!
Love!
Grades!
**Love your structs!**

const int NameLen = 20;

typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

*Is this legal?* x.shelfArr.bookArr.idnum = 23;
const int NameLen = 20;
typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

- Is this legal?
  x.shelfArr.bookArr.idnum = 23;

- No!
- Both bookArr and shelfArr are arrays.
- This code attempts to access members of pointer values...
const int NameLen = 20;
typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

- **We fix it:**

- **Now we have subscripted those arrays -> Accessed what they are pointing to!**

- **Recommended:** writing more abstract data access functions
const int NameLen = 20;

typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

• So how about...
  p.numOfShelves = 4
const int NameLen = 20;

typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

- So how about...
  p.numOfShelves = 4;
- No!
- p is a pointer value.
- It isn't a Library value.
Love your structs!

```c
const int NameLen = 20;
typedef struct {
    char bookName[NameLen];
    int idnum;
} Book;

typedef struct {
    Book bookArr[NameLen];
    int arrSize;
} Shelf;

typedef struct {
    char libName[NameLen];
    Shelf shelfArr[NameLen];
    int numOfShelves;
} Library;

Library x;
Library *p;

● **We fix it:**
   p->numOfShelves = 4;

● **You need to use -> to dereference a pointer to a struct so you can get to its members.**

● **Same as (and neater than) saying:**
   (*p).numOfShelves = 4;
```
Write functions that love your structs!

- Write a function that takes a Book and prints its contents on a single line followed by an end-of-line char.
- Takes a Shelf and (using the previous), prints out all of the Books on the Shelf.
- Takes a Library and prints its name on one line and then prints out all the shelves.
- Takes two books and prints their titles in order of increasing idnum.

No-Heart does not want you to write these.
A love note for unions

- **Example**
  ```c
  typedef union {
    int quantity;
    float quality;
  } Chocolate;
  ```

- **This is not a struct**: you can only use one field at a time. (ie, you can either have quantity or quality but not both.)

- **Stored in same memory**: C will not prevent you from looking at both. But float memory is different from int memory.

- **Why use them?** Same var, different actual types... Sound familiar?
Next lab, we return to Evil Robots