

# Structures III

CS2263 – Systems Software Development

1

## Learning Outcomes

At the conclusion of this lecture students should be able to:

- Explain how the examples here “work” in terms of memory use and functionality.
- Extend the examples here to include abstracted input/output functionality

2

## References

- Modules in C.  
[www.cs.princeton.edu/courses/archive/spring03/cs217/lectures/Modules.pdf](http://www.cs.princeton.edu/courses/archive/spring03/cs217/lectures/Modules.pdf) Notes from Princeton's CS 217
- Steve Oualline. 2011. Practical C programming. O'Reilly Media.  
*Available as an ebook from UNB Libraries.*

3

## Review[o]

- What is an array?
- What is a structure?
- How are they "laid out" in memory?
  - Stack
  - Heap
- How do pointers help?
- Pointer access of each

4

## Review [1]

- Abstraction/encapsulation in C
  - Structures
  - Functions
  - Headers
  - Preprocessor
  - Pointers/Dynamic memory

5

## Review [2]

- Example: Point2D module
  - Structure
  - Functions
  - Headers
  - Preprocessor
  - Pointers/Dynamic memory

6

## Review [3]

- Extended Example: StringList (aka array of Strings, aka char\*\*)
  - Structure
  - Functions
  - Headers
  - Preprocessor
  - Pointers/Dynamic memory

7

## Extending the idea: I/O

- Alter the modules (Point2D and StringList) abstract away I/O
  - What should the function interfaces look like?
  - Format for the structures as a fragment in a file (serialization)?

8

## Define these Functions

```
pPoint2D fscanfPoint2D(FILE* pF);  
// pPoint2D freadPoint2D(FILE* pF); <- read from a binary file  
String fscanfString(FILE* pF);  
pLabel2D fscanfLabel2D(FILE* pF);  
pLine2D fscanfLine2D(FILE* pF); // if it's an array of Point2D
```