Structures

Struct.c: Structure and Pointers at Depth

- Passing of data (pass by reference)
- Arrays of Arrays
- Grouping
- Simplicity
- C & C++

Members

- Should be constant
- Char quantity [131]
- Char description (131)
- Char name [21]
- Int board [13]
- Float price [3]
Inventory

Vocabulary of these

0. Number

all-product: Ci]

for (int c = ... cost <= end

cost << all-products price]

for (int c = ... cost >> all-products

name

for (int c = ... c < num-product?

int num-product = 0;

Inventory all-product [100]
Load-inventory (all-products, num-product)

if num > 100
  exit

while (response != "n")
  cin >> response
  count ++
  if count >= 15 then another product?";
  read-inventory (array [num]);
  do
    if num > 100 return!
    cin >> response;
  end
  end

load-inventory (Inventory_array)
- Just group data
- In C Language

```
struct {
  struct "publite"
  struct "struct"
} public
```

- Create operations that work on the data
- Grouping of data types
- Class "class"
CS 202
CS 163

Object-oriented Programming

Object

Abstract Data Types

Operations

Pettery (word)

INT

Play in the class

Class — ADT — OOP
A product

Data inside

"Job" of a product

What should the

be

public

class product

Tag name

Raw name

Unique Func

Data Manages

Private:

Mutily Funcs

Platform:

Data that is

represented

using classes

accessible

to programs

Functions
```cpp
{ 
  cin >> price; cin.ignore (c);
  cin.ignore (100, 'n'); cin >> "Enter Price ";
  cost >> "Please enter product name";
  product :: read (c);
  
  distrubutor [phi] = \phi;
  balance [phi] = \phi;
  price = \phi; phi;
  name [phi] = \phi;
}
```
```
// Create (variable) or object of the class

// Get (with name)

// Set

// Construct

// Invoke
```

```
// Product Milk!

// Object

// Create Display (C)

// Milk Read (C)

// Object
```

```
```

```
```

```
```
any \{ \text{num-prod} \otimes \text{index} \}\}

\text{void good-\text{inventory} (product array)}

\text{void good-\text{inventory} (all, num-prod)}

\text{int num-prod} = \Phi\}

\text{product all} \subseteq \mathbb{S}_{\text{EE}}\}
def copy(yp):
    for product in all_products[num-1]:
        if num-1 > size-1:
            if error == 0:
                
                int start_inventory == add item (product 8p)

            
        return error

        error = 1

        num-1 = 0

        claimant

        start_inventory == start inventory (at size 0)
In the `class
\{ all-products [1]. display();
    number
    \}
for (int i = 0; i < num-jpd; i++)
    print-to-screen(); display();
int
Your programs

#include
class interfaces
prototypes

main
implementation of member functions

any order

I file
# include "inv.h"

```
#include "inv.h"

// include "inv.h"
```

```
#include "inv.h"
```

```
```