

# Computer Programming

Dr. Deepak B Phatak  
Dr. Supratik Chakraborty  
Department of Computer Science and Engineering  
IIT Bombay

Session: Programming using structures – Part 1

# Quick Recap of Relevant Topics

---



- Brief introduction to object-oriented programming
- Defining structures in C++
- Accessing members of structures
- Initializing and copying structures

# Overview of This Lecture

---



- Getting our hands dirty
  - C++ programming with structures

# Acknowledgment

---



- Some examples in this lecture are from  
**An Introduction to Programming Through C++**  
by Abhiram G. Ranade  
McGraw Hill Education 2014
- All such examples indicated in slides with the citation  
**AGRBook**

# Recall: Library Information Management System

## [Ref. AGRBook]

---



- Every patron has a numerical id
- Every book has an accession number
- **Check out:** A patron can check out upto 3 books at any time
- **Claim:** If X has not already checked out 3 books, she can claim a book checked out by Y  
When Y returns the book, it is held for X and cannot be lent to others
- **Return:** A patron can return a book checked out by her at any time

# Recall: Relevant Structures and Arrays

```
struct Book {  
    char title[50];  
    char authors[500];  
    double price;  
    int accNum;  
    bool checkOutStatus;  
    int claimantId;  
};
```

**Book** libraryShelf[1000]

# Recall: Relevant Structures and Arrays

```
struct Book {  
    char title[50];  
    char authors[500];  
    double price;  
    int accNum;  
    bool checkOutStatus;  
    int claimantId;  
};
```

**Assume checkOutStatus and claimantId of all elements of array libraryShelf initialized to “false” and “-1” respectively**

**Book** libraryShelf[1000]

# Recall: Relevant Structures and Arrays



```
struct Patron {  
    char name[50];  
    char address[100];  
    int uniqueId;  
    int numBooksChkdOut;  
    int claimdBookAccNum;  
};
```

**Patron** libraryPatrons[200]



# Recall: Relevant Structures and Arrays

**Assume numBooksChkdOut and claimdBookAccNum for all elements of array libraryPatrons initialized to "0" and "-1" respectively**

```
struct Patron {  
    char name[50];  
    char address[100];  
    int uniqueId;  
    int numBooksChkdOut;  
    int claimdBookAccNum;  
};
```

**Patron** libraryPatrons[200]

# C++ Function for Checking Out a Book



`currPatron` (of type **Patron**) to check out `currBook` (of type **Book**)

- Check if `currPatron` has already checked out 3 books
  - If so, print appropriate message and return
  - Otherwise,
    - \* If `currBook` is already checked out, print appropriate message and return
    - \* Otherwise, if `currBook` not already claimed by a different patron
      - Increment value of `currPatron.numBooksChkdOut`
      - Set `currBook.checkOutStatus` of book to true

# C++ Function for Checking Out a Book



**We want changes to members of currPatron and currBook to persist after currBook is checked out**  
**Need functions with parameters passed by reference**

- \* Otherwise, if currBook is not already claimed by a different patron
  - Increment value of currPatron.numBooksChkdOut
  - Set currBook.checkOutStatus of book to true

# C++ Function for Checking Out a Book

```
void checkOutBook(Patron &currPatron, Book &currBook)
{
    ... Code for checking out a book ...
}

int main() {
    ... Other code ...
    checkoutBook(libraryPatrons[i], libraryShelf[j]);
    ... Other code ...
}
```

# C++ Function for Checking Out a Book



```
// PRECONDITION: Members of currPatron and currBook are  
//                properly initialized – no garbage values
```

```
void checkOutBook(Patron &currPatron, Book &currBook)  
{
```

```
    ... Code for checking out a book ...
```

```
}
```

```
// POSTCONDITION: If currBook is lent to currPatron, members  
// currPatron and currBook appropriately updated
```

# C++ Function for Checking Out a Book

```
void checkOutBook(Patron &currPatron, Book &currBook)
```

```
{
```

```
    if (currPatron.numBooksChkdOut < 3) {
```

```
        ... Code for checking out a book (part 1) ...
```

```
    }
```

```
    else { cout << "Sorry! Three books have already been checked";
```

```
        cout << " out by " << currPatron.name << endl;
```

```
        return;
```

```
    }
```

```
}
```

# C++ Function for Checking Out a Book

```
if (currPatron.numBooksChkdOut < 3) {  
    if (currBook.checkOutStatus == true) {  
        cout << "Sorry! Book " << currBook.title;  
        cout << " (Accession # " << currBook.accNum << ") ";  
        cout << " already checked out!" << endl;  
        return;  
    }  
    else { ... Code for checking out a book (part 2) ... }  
}
```

# C++ Function for Checking Out a Book

```
else {  
    if ((currBook.claimantId != -1) &&  
        (currBook.claimantId != currPatron.uniqueId)) {  
        cout << "Sorry! There is already a pending claim on book ";  
        cout << currBook.title << " (Acc # " << currBook.accNum << ") ";  
        cout << " by a different patron." << endl;  
        return;  
    }  
    else { ... Code for checking out a book (part 3) ... }  
}
```



# C++ Function for Checking Out a Book

```
else {  
    currBook.checkOutStatus = true;  
    currPatron.numBooksChkdOut ++;  
    if (currBook.claimantId ==  
        currPatron.uniqueId) {  
        currPatron.claimdBookAccNum = -1;  
        currBook.claimantId = -1;  
    }  
    return;  
}
```

**currBook, currPatron  
passed by reference**

**Therefore, structures  
used as parameters in  
the calling function are  
updated**

# Summary

---



- Programming using structures
  - Passing structures as parameters to functions
  - Accessing members of structures in program

**We aren't done yet with our implementation of different functions needed in the library information management system. Subsequent lecture to cover these.**