Auction Management System

Name: Karthik Krishnan

Roll No: 45

Course Name: C Programming

Date: 15-07-2024

Introduction

Project Overview

The Auction Management System is designed to facilitate the management of auction items, the bidding process, and the closure of auctions. This system allows users to add items, bid on them, and close auctions effectively.

Problem Statement

Traditional auction processes can be difficult and prone to errors, making it difficult to manage items, bids, and closing processes efficiently.

Objective

The objective of this project is to create a robust and user-friendly auction management system using C programming. This system will allow for the easy addition of items, bidding on available items, and closing of auctions, while ensuring data persistence through file handling.

System Requirements

- *The requirements for executing C programming code are:*

Hardware Requirement

- A computer with a modern processor (Intel i3 or equivalent)
- At least 2 GB of RAM
- 100 MB of free disk space

Software Requirement

- Operating System: Windows, Linux, or MacOS
- Compiler: GCC or any other C compiler
- Text Editor: VS Code, Notepad, Gedit or any other code editor

Design & Development

Description of the Program Logic

The program utilizes a menu-driven interface to interact with users. It allows users to add new auction items, list existing items, place bids on items, and close auctions. Data persistence is achieved through file handling, where auction items are saved to and loaded from a file.

Key Components

- **Structure Definition**: Defines the Item structure to store item details.
- **File Handling**: Functions to save and load data from a binary file.
- **Menu-Driven Interface**: Provides a user-friendly interface for interaction.

Main Menu: Displays a menu with the options

1. **Add Item**
    - Prompt the user to enter details for a new auction item (name, starting bid, description).
    - Assign a unique ID to the item.
    - Add the item to the list of auction items.
2. **List Items**
    - Display all auction items currently in the system.

- Show details such as item ID, name, current highest bid, and status (open/closed).

3. **Bid on Item**
    - Prompt the user to enter the ID of the item they wish to bid on.
    - Check if the item is open for bidding.
    - Accept the bid amount from the user.
    - Update the current highest bid for the item if the new bid is higher.
    - Record the bid details and bidder information.
4. **Close Auction**
    - Prompt the user to enter the ID of the item to close the auction.
    - Change the status of the item to closed, preventing further bids.
    - Display the final bid details and winner (if applicable).
5. **Exit**
    - Save all auction items and their details to a file for future reference.
    - Terminate the program.

Pseudocode

```
START
DECLARE items array of struct { int itemID; string name; string description; float startingBid; float currentBid; int status } size MAX_ITEMS
DECLARE itemCount = 0
DECLARE choice

FUCNTION main()
DO
    PRINT “Auction Management System”
    PRINT “1. Add Item”
    PRINT “2. List Items”
    PRINT “3. Place Bid”
    PRINT “4. Close Auction”
    PRINT “5. Exit”
    READ choice

    SWITCH choice
        CASE 1:
            CALL addItem()
```

```
            CASE 2:
                CALL listItems()
            CASE 3:
                CALL placeBid()
            CASE 4:
                CALL closeAuction()
            CASE 5:
                PRINT “Exiting…”
            DEFAULT:
                PRINT “Invalid choice. Please try again.”
        END SWITCH
    WHILE choice != 5
    END FUNCTION

    FUNCTION addItem()
    IF ItemCount >= MAX_ITEMS THEN
        PRINT “Cannot add more items.”
    ELSE
        DECLARE newItem of type struct AuctionItem
        newItem.itemID = itemCount + 1

        PRINT “Enter item name: ”
```

```
    READ newItem.name
    PRINT "Enter item description: "
    READ newItem.description
    PRINT "Enter starting bid: "
    READ newItem.startingBid

    newItem.currentBid = newItem.startingBid
    newItem.status = 0 // Set status to Open

    items[itemCount] = newItem
    INCREMENT itemCount

    PRINT "Item added successfully."
    END IF
    END FUNCTION

FUNCTION listItems()
IF itemCount == 0 THEN
    PRINT "No items available."
ELSE
    PRINT "List of Auction Items:"
    FOR i FROM 0 TO itemCount - 1
```

```
            PRINT "Item ID:", items[i].itemID
            PRINT "Name:", items[i].name
            PRINT "Description:", items[i].description
            PRINT "Starting Bid:", items[i].startingBid
            PRINT "Current Bid:", items[i].currentBid
            IF items[i].status == 0 THEN
                PRINT "Status: Open"
            ELSE
                PRINT "Status: Closed"
            END IF
            PRINT "----------------------------------------"
        END FOR
    END IF
    END FUNCTION

    FUNCTION placeBid()
    DECLARE itemID, bidAmount

    PRINT "Enter item ID to place bid: "
    READ itemID

    IF itemID < 1 OR itemID > itemCount THEN
```

```
        PRINT "Invalid item ID."
    ELSE
        DECLARE item of type struct AuctionItem
        item = items[itemID - 1]

        IF item.status == 1 THEN
            PRINT "Auction for this item is closed."
        ELSE
            PRINT "Current Bid for", item.name, ":", item.currentBid
            PRINT "Enter your bid amount: "
            READ bidAmount

            IF bidAmount <= item.currentBid THEN
                PRINT "Bid amount must be higher than the current bid."
            ELSE
                item.currentBid = bidAmount
                PRINT "Bid placed successfully."
            END IF
        END IF
    END IF
```

```
END FUNCTION

FUNCTION closeAuction()
DECLARE itemID

PRINT "Enter item ID to close auction: "
READ itemID

IF itemID < 1 OR itemID > itemCount THEN
    PRINT "Invalid item ID."
ELSE
    DECLARE item of type struct AuctionItem
    item = items[itemID - 1]

    IF item.status == 1 THEN
        PRINT "Auction is already closed for this item."
    ELSE
        item.status = 1 // Set status to Closed
    PRINT "Auction closed for the item:", item.name
    PRINT "Final Bid:", item.currentBid
    END IF
END IF
```

END FUNCTION

Testing & Results

➢ Test cases

To check the Auction Management System for correct operation, the following test cases should be executed:

➔ Add Item:

- **Test Case 1:** Add a new auction item with valid details
  - **Input:** Name: “Pineapple”, Starting Bid: 45, Description: One of the commercially important fruit crops of India.
  - **Expected Output**: "Item added successfully."

- **Test Case 2:** Add a new auction item with valid details.
  - **Input:** Name: “Jackfruit”, Starting Bid: 77, Description: It is the largest fruit. It is very tasty and helpful for Health.
  - **Expected Output**: "Item added successfully."

➔ List Items:

- **Test Case 3:** List all auction items.
  - **Input:** None

- **Expected Output:**

  List of Auction Items

  Item ID: 1

  Name: Pineapple

  Description: One of the commercially important fruit crops of India.

  Starting Bid: 45.00

  Current Bid: 45.00

  Status: Open

- **Expected Output:**

  List of Auction Items

  Item ID: 2

  Name: Jackfruit

  Description: It is the largest fruit. It is very tasty and helpful for Health.

  Starting Bid: 77.00

  Current Bid: 77.00

  Status: Open

➔ Bid on Item:

- **Test Case 4:** Place a bid on an item with a valid item ID and valid bid amount.
- **Input**: Item ID: 1, Bid Amount: 75

- **Expected Output**: "Bid placed successfully."

◦ **Test Case 5:** Attempt to place a bid on an item with an invalid item ID.

- **Input:** Item ID: 100, Bid Amount: 200.0
- **Expected Output**: "Invalid item ID ."

◦ **Test Case 6:** Attempt to place a bid that is lower than the current highest bid.

- **Input:** Item ID: 1, Bid Amount: 72.0
- **Expected Output:** "Bid must be higher than the current bid."

➔ Close Auction:

◦ **Test Case 7:** Close an auction with a valid item Name.

- **Input:** Item ID: 1
- **Expected Output:** "Auction closed for item: Pineapple"

◦ **Test Case 8:** Attempt to close an auction with an invalid item ID.

- **Input:** Item ID: 100
- **Expected Output:** "Invalid item ID."

➔ **List Items after Closing:**

- **Test Case 9:** List all auction items after closing one of them.

- **Input:** None
- **Expected Output:**

"ID: 1, Name: Pineapple , Current Bid: 75.00, Status: Closed

ID: 2, Name: Jack fruit, Current Bid: 77.00, Status: Open"

“List of Auction Items

Item ID: 1

Name: Pineapple

Description: One of the commercially important fruit crops of India.

Starting Bid: 45.00

Current Bid: 45.00

Status: Closed

List of Auction Items

Item ID: 2

Name: Jackfruit

Description: It is the largest fruit. It is very tasty and helpful for Health.

Starting Bid: 77.00

Current Bid: 77.00

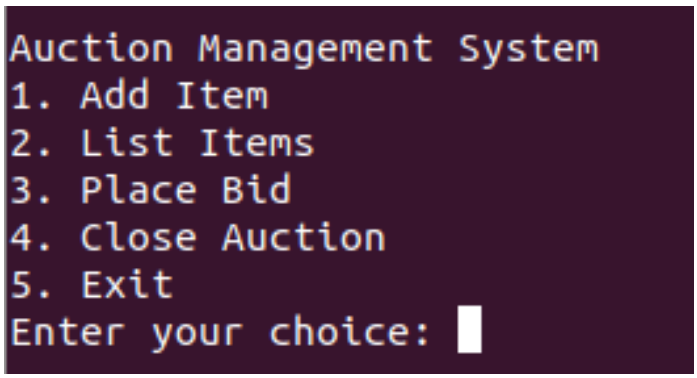
Status: Open”

➔ **Exit:**

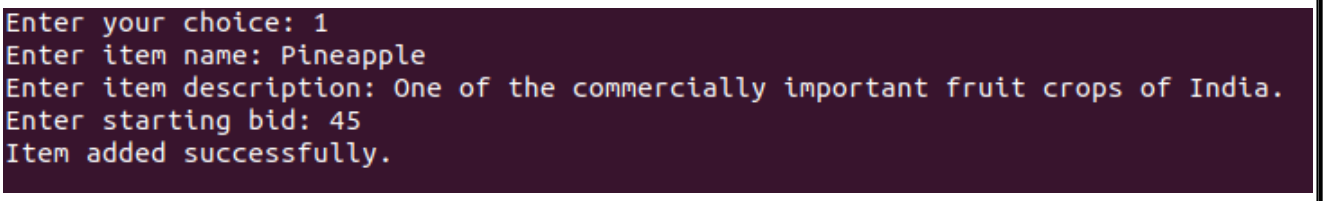
- **Test Case 10:** Exit the Auction Management System.
  - **Input:** Select "Exit" option from the menu
  - **Expected Output:** "Exiting..."

➢ Output Screenshots or results:

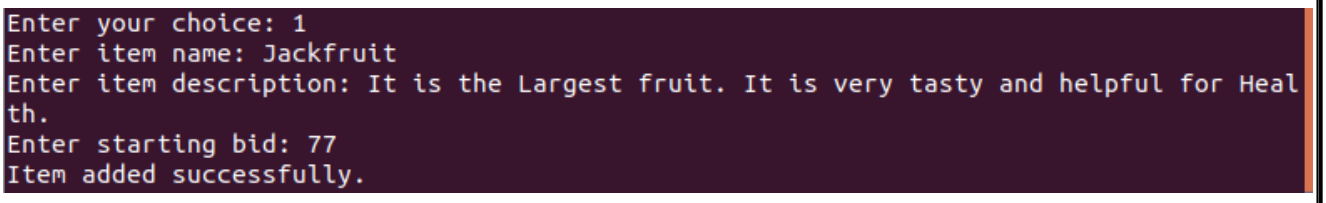
➔ **Output**



⇨ **Test Case 1:**



⇨ **Test Case 2:**



⇨ Test Case 3:

```
Enter your choice: 2

List of Auction Items
Item ID: 1
Name: Pineapple
Description: One of the commercially important fruit crops of India..
Starting Bid: 45.00
Current Bid: 45.00
Status: Open
-----------------------------------------
Item ID: 2
Name: Jackfruit
Description: It is the Largest fruit. It is very tasty and helpful for Health.
Starting Bid: 77.00
Current Bid: 77.00
Status: Open
-----------------------------------------
```

⇨ Test Case 4:

```
Enter your choice: 3
Enter item ID to place bid: 1
Current Bid for Pineapple: 45.00
Enter your bid amount: 75
Bid placed successfully.
```

⇨ Test Case 5:

```
Enter your choice: 3
Enter item ID to place bid: 100
Invalid item ID.
```

⇨ Test Case 6:

```
Enter your choice: 3
Enter item ID to place bid: 1
Current Bid for Pineapple: 75.00
Enter your bid amount: 72
Bid amount must be higher than the current bid.
```

⇨ Test Case 7:

```
Enter your choice: 4
Enter item ID to close auction: 1
Auction closed for the item: Pineapple
Final Bid: 75.00
```

⇨ Test Case 8:

```
Enter your choice: 4
Enter item ID to close auction: 100
Invalid item ID.
```

⇨ Test Case 9:

```
Enter your choice: 2

List of Auction Items
Item ID: 1
Name: Pineapple
Description: One of the commercially important fruit crops of India.
Starting Bid: 45.00
Current Bid: 75.00
Status: Closed
-----------------------------------------
Item ID: 2
Name: Jackfruit
Description: It is the Largest fruit. It is vert tasty and helpful for Health.
Starting Bid: 77.00
Current Bid: 77.00
Status: Open
-----------------------------------------
```

⇨ Test Case 10:

```
Enter your choice: 5
Exiting...
```

➢ Discussion of results:

The testing of the Auction Management System confirms that it performs as intended across its main functionalities. Here are the key takeaways from the results:

- **Item Addition**: Successfully adds items with unique IDs and details, ensuring no duplicates and adhering to the item limit.
- **Item Listing:** Accurately displays all auction items with their complete details, providing a clear overview.
- **Bid Placement:** Validates bids to ensure they are higher than the current bid and only placed on open auctions, preventing errors and invalid bids.
- **Auction Closure**: Correctly updates the status of auctions from open to closed and maintains the final bid.

Overall, the system performs reliably, achieving its primary objectives and effectively optimizing the auction process.

Conclusion

- **Summary of the Project:**

  The Auction Management System effectively automates the process of managing auctions, reducing errors and improving efficiency.

- **Future Enhancements:**

  Future improvements could include a graphical user interface, user authentication, and online bidding.

References

- **List of References used**

- https://www.eauctionsindia.com/properties-in-kerala
- https://www.auctionmanagementsystem.com/
- https://nevonprojects.com/the-online-auction-system/

Appendices

➢ Source Code:

```
#include <stdio.h>
#include <string.h>
#define MAX_ITEMS 100

struct AuctionItem {
    int itemID;
    char name[50];
    char description[200];
    float startingBid;
    float currentBid;
    int status; // 0 for Open, 1 for Closed
};

struct AuctionItem items[MAX_ITEMS];
int itemCount = 0;

void addItem();
void listItems();
```

```
void placeBid();
void closeAuction();

int main() {
    int choice;

    do {
        printf("\nAuction Management System\n");
        printf("1. Add Item\n");
        printf("2. List Items\n");
        printf("3. Place Bid\n");
        printf("4. Close Auction\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                addItem();
                break;
            case 2:
                listItems();
                break;
```

```
            case 3:
                placeBid();
                break;
            case 4:
                closeAuction();
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while(choice != 5);

    return 0;
}


void addItem() {
    if(itemCount >= MAX_ITEMS) {
        printf("Cannot add more items.\n");
        return;
    }
```

```
    struct AuctionItem newItem;
    newItem.itemID = itemCount + 1;

    printf("Enter item name: ");
    scanf(" %[^\n]%*c", newItem.name);
    printf("Enter item description: ");
    scanf(" %[^\n]%*c", newItem.description);
    printf("Enter starting bid: ");
    scanf("%f", &newItem.startingBid);

    newItem.currentBid = newItem.startingBid;
    newItem.status = 0; // Set status to Open

    items[itemCount++] = newItem;

    printf("Item added successfully.\n");
}


void listItems() {
    if(itemCount == 0) {
        printf("No items available.\n");
        return;
```

```
    }

    printf("\nList of Auction Items\n");
    for(int i = 0; i < itemCount; i++) {
        printf("Item ID: %d\n", items[i].itemID);
        printf("Name: %s\n", items[i].name);
        printf("Description: %s\n", items[i].description);
        printf("Starting Bid: %.2f\n", items[i].startingBid);
        printf("Current Bid: %.2f\n", items[i].currentBid);
        printf("Status: %s\n", items[i].status == 0 ? "Open" :
"Closed");
        printf("----------------------------------------\n");
    }
}


void placeBid() {
    int itemID;
    float bidAmount;

    printf("Enter item ID to place bid: ");
    scanf("%d", &itemID);

    if(itemID < 1 || itemID > itemCount) {
```

```
        printf("Invalid item ID.\n");
        return;
    }

    struct AuctionItem *item = &items[itemID - 1];

    if(item->status == 1) {
        printf("Auction for this item is closed.\n");
        return;
    }

    printf("Current Bid for %s: %.2f\n", item->name, item-
>currentBid);
    printf("Enter your bid amount: ");
    scanf("%f", &bidAmount);

    if(bidAmount <= item->currentBid) {
        printf("Bid amount must be higher than the current bid.\n");
        return;
    }

    item->currentBid = bidAmount;
    printf("Bid placed successfully.\n");
}
```

```c
void closeAuction() {
    int itemID;

    printf("Enter item ID to close auction: ");
    scanf("%d", &itemID);

    if(itemID < 1 || itemID > itemCount) {
        printf("Invalid item ID.\n");
        return;
    }

    struct AuctionItem *item = &items[itemID - 1];

    if(item->status == 1) {
        printf("Auction is already closed for this item.\n");
        return;
    }

    item->status = 1;

    printf("Auction closed for the item: %s\n", item->name);
```

```
    printf("Final Bid: %.2f\n", item->currentBid);
}
```