ABSTRACT

The "Medical Store Management System" is a simplified mini project implemented in the C programming language. The program aims to provide a basic console-based application for managing medicines in a medical store. It allows users to add new medicines,remove medicine, display existing medicines and generate bill based on the given stock and prices.

Firstly users can add new medicines by entering the medicine name. The program ensures that the maximum limit of medicines is not exceeded.If the user wants to remove medicine it can be removed by entering the medicine name.finally bill can be generated based on the quantity.

This project serves as an introductory example for learning C programming concepts, including structures, arrays, loops, functions, and user input handling. However, it is essential to understand that a comprehensive medical store management system in the real world would require more sophisticated features such as data storage, sales history tracking, billing, and inventory management.

The "Medical Store Management System" provides an excellent starting point for students and beginners to explore the fundamentals of C programming in the context of a medical store scenario.

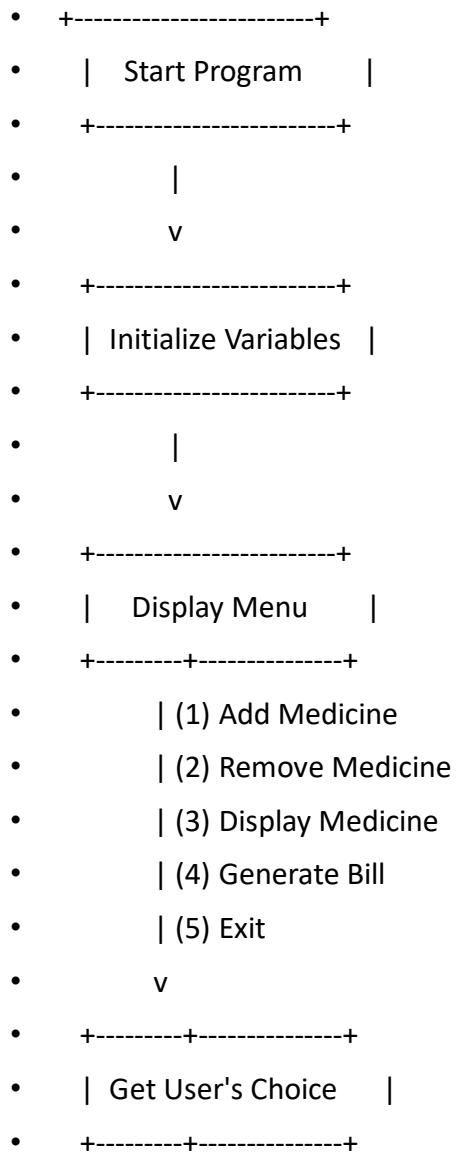
INTRODUCTION:

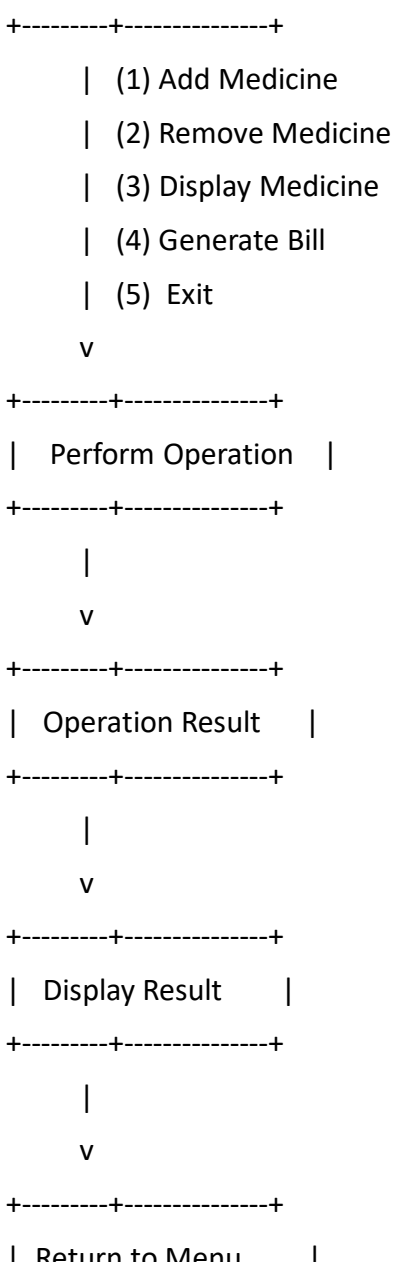
- The primary objective of this project is to provide a simple yet educational example for students and beginners to learn the fundamental concepts of C programming. By creating a medical store management system, learners can gain insights into handling structures, arrays, functions, loops, and user input in a real-world scenario.
- The program allows the user to add new medicines to the medical store by providing details such as the medicine name, stock quantity, and price. It ensures that the number of medicines does not exceed the maximum limit.
- The system can display a list of medicines available in the store. The information displayed includes the medicine name, price,quantity and bill for viewing.
- This program can serve as a solid foundation for students to expand their programming skills and explore more complex projects in the domain of inventory management, billing, and sales tracking for medical stores. It is crucial to emphasize the importance of continuous learning and exploration beyond this basic implementation to build practical and efficient medical store management systems.

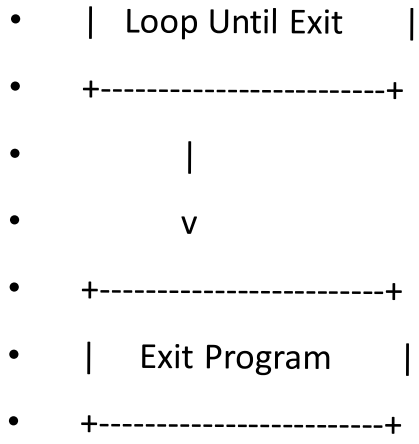
SCOPE:

- The scope of the "Medical Store Management System" implemented in C programming includes the following aspects:
- Medicine management:
- Users can add medicine
- User can remove medicine
- User can display medicine
- The bill can be generated based on the given stock and price
- Limited Real-World Usability: As a basic mini project, this implementation lacks many advanced features needed in a fully functional medical store management system. It does not include data persistence, advanced inventory management, sales history tracking, or security measures.
- Simplified Implementation: The project provides a simplified approach to medical store management and serves as a stepping stone for learners to explore more advanced projects in the field.

DESIGN:


Start Program
Initialize Variables
Display Menu
(1) Add Medicine
(2) Remove Medicine
(3) Display Medicine
(4) Generate Bill
(5) Exit
Get User's Choice


+---------+---------------+
| (1) Add Medicine
| (2) Remove Medicine
| (3) Display Medicine
| (4) Generate Bill
| (5) Exit
v
+---------+---------------+
| Perform Operation |
+---------+---------------+
|
v
+---------+---------------+
| Operation Result |
+---------+---------------+
|
v
+---------+---------------+
| Display Result |
+---------+---------------+
|
v
+---------+---------------+
| Return to Menu |


| Loop Until Exit |
+-------------------------+
|
v
+-------------------------+
| Exit Program |
+-------------------------+

SOURCE CODE:

```
#include <stdio.h>

#include <string.h>

// Define a structure to represent medicine

typedef struct {

    char name[100];

    float price;

    // Add more fields if needed (e.g., quantity, batch number, etc.)

} Medicine;


#define MAX_MEDICINES 100

// Function prototypes

void addMedicine(Medicine[], int*);

void removeMedicine(Medicine[], int*);
```

```
void displayMedicines(Medicine[], int);
void generateBill(Medicine[], int);

int main() {
    Medicine medicines[MAX_MEDICINES];
    int numMedicines = 0;
    char choice;

    do {
        // Display menu
        printf("\nMedical Store Management System\n");
        printf("1. Add Medicine\n");
        printf("2. Remove Medicine\n");
```

```
printf("3. Display Medicines\n");
        printf("4. Generate Bill\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf(" %c", &choice);

        switch (choice) {
            case '1':
                addMedicine(medicines, &numMedicines);
                break;
            case '2':
                removeMedicine(medicines, &numMedicines);
                break;
```

```
case '3':
            displayMedicines(medicines, numMedicines);
            break;
        case '4':
            generateBill(medicines, numMedicines);
            break;
        case '5':
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
```

```
    } while (choice != '5');

    return 0;
}

void addMedicine(Medicine medicines[], int *numMedicines) {
    if (*numMedicines == MAX_MEDICINES) {
        printf("Medicine database is full. Cannot add more medicines.\n");
        return;
    }

    printf("Enter medicine name: ");
    scanf(" %[^\n]", medicines[*numMedicines].name);
```

```
printf("Enter medicine price: ");
    scanf("%f", &medicines[*numMedicines].price);

    (*numMedicines)++;
    printf("Medicine added successfully.\n");
}

void removeMedicine(Medicine medicines[], int *numMedicines) {
    if (*numMedicines == 0) {
        printf("No medicines in the database to remove.\n");
        return;
    }
```

```
char searchName[100];
    printf("Enter the name of the medicine to remove: ");
    scanf(" %[^\n]", searchName);


    int i, found = 0;
    for (i = 0; i < *numMedicines; i++) {
        if (strcmp(medicines[i].name, searchName) == 0) {
            found = 1;
            break;
        }
    }
```

```
    if (found) {
        // Shift the array elements to remove the medicine
        for (int j = i; j < (*numMedicines) - 1; j++) {
            medicines[j] = medicines[j + 1];
        }

        (*numMedicines)--;
        printf("Medicine removed successfully.\n");
    } else {
        printf("Medicine not found in the database.\n");
    }
}
```

```
void displayMedicines(Medicine medicines[], int numMedicines) {
    if (numMedicines == 0) {
        printf("No medicines in the database.\n");
        return;
    }

    printf("Medicine List:\n");
    for (int i = 0; i < numMedicines; i++) {
        printf("%d. Name: %s, Price: %.2f\n", i + 1, medicines[i].name,
medicines[i].price);
    }
}
```

```
void generateBill(Medicine medicines[], int numMedicines) {
    if (numMedicines == 0) {
        printf("No medicines in the database to generate a
bill.\n");
        return;
    }

    float total = 0;
    char searchName[100];
    int quantity;

    printf("Enter medicine name to add to the bill: ");
    scanf(" %[^\n]", searchName);
```

```
int i, found = 0;
    for (i = 0; i < numMedicines; i++) {
        if (strcmp(medicines[i].name, searchName) ==
0) {
            found = 1;
            break;
        }
    }

    if (found) {
        printf("Enter the quantity of medicine: ");
        scanf("%d", &quantity);
```

```
total = medicines[i].price * quantity;
        printf("Medicine added to the bill.\n");
        printf("Total Bill: %.2f\n", total);
    } else {
        printf("Medicine not found in the
database.\n");
    }
}
```

OUTPUT:


main.c
Output
Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 1
Enter medicine name: amaxillin
Enter medicine price: 5
Medicine added successfully.
Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 1
Enter medicine name: colpol
Enter medicine price: 2
Medicine added successfully.
Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit

```
Enter your choice: 1
Enter medicine name: erythromycin
Enter medicine price: 6
Medicine added successfully.

Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 2
Enter the name of the medicine to remove:
     erythromycin
Medicine removed successfully.

Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 3
Medicine List:
1. Name: amaxillin, Price: 5.00
2. Name: colpol, Price: 2.00

Medical Store Management System
```

```
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 4
Enter medicine name to add to the bill:
    amaxillin
Enter the quantity of medicine: 6
Medicine added to the bill.
Total Bill: 30.00

Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
4. Generate Bill
5. Exit
Enter your choice: 4
Enter medicine name to add to the bill:
    colpol
Enter the quantity of medicine: 12
Medicine added to the bill.
Total Bill: 24.00
Medical Store Management System
1. Add Medicine
2. Remove Medicine
3. Display Medicines
```

```
4. Generate Bill
5. Exit
Enter your choice: 5
Exiting the program.
```

CONCLUSION:

- In conclusion, the "Medical Store Management System" implemented in C programming offers a basic yet functional console-based application to manage medicines in a medical store. The program allows users to add new medicines, display existing medicines with their stock and prices, calculate total sales, and exit the program.
- The system provides essential features for medical store management, allowing users to add medicines to the store's inventory, view the available medicines, and calculate the total sales based on the available stock and prices.
- The "Medical Store Management System" project provides a foundation for students and beginners to build their programming skills and explore more complex projects related to inventory management and sales tracking for medical stores.

REFERENCE:

- Reference links
- 1. https://www.iso.org/standard/74528.html
- 2. https://www.tutorialspoint.com/cprogramming/index.htm