# VOTING SYSTEM

Name : Arjun Santhosh E N

Roll No. : 19

Course Name : C Programming

Date : 10 - 07 – 24

# Introduction

- ## Project overview :

The project entails the development of a robust voting system that ensures secure vote counting, accurate voter registration, and transparent display of votes received by each candidate. The system will be implemented using the C programming language, emphasizing security and efficiency.

## Problem statement :

The current voting systems are vulnerable to security breaches, allow for the possibility of people voting more than once, and often have mistakes in the voter registration records. These problems can lead to election fraud and result in incorrect election results.

- ## Objective :

Create a voting system that can conduct elections or polls with a secure method for registering voters, ensuring that each voter is uniquely identified, to count votes securely, Prevent Double Voting, a feature to transparently display the number of votes each candidate has received.

# System Requirements

## ❖ Minimum Requirements for C Programming Code to Run:

### ➢ Hardware Requirement:

- **Processor**: Any processor with at least 1 GHz speed
- **Storage**: Minimum 1GB free disk space
- **RAM**: Minimum 2GB (4GB recommended)

### ➢ Software Requirement:

- **Operating System**: Windows 7 or higher (Windows 10 recommended)
- **Compiler**: GCC or any compatible C compiler
- **IDE**: Any lightweight C IDE like Visual Studio, code blocks

# Design and Development

## ❖ Program Logic :

➲ **Initialization**:

- Structs voter and candidates are defined to store voter details and candidate information.
- Initial candidates are present with their IDs, names, and have 0 votes.

➲ **Main Menu** : Displays a menu with following options :

1. **Register Voter** :

   - Prompts the user to enter their name.
   - Assigns a unique voter ID and marks them as not voted.
   - Increments the total number of registered voters .

2. **Vote** :

   - Prompts the user to enter their voter ID.
   - Checks if the voter is registered and hasn't already voted.
   - Displays a list of candidates and allows the voter to choose one.
   - Records the vote, increments the candidate's vote count, and marks the voter as voted.

3. **Vote Count** :

   - Displays the total votes received by each candidate.

4. **Exit**:

   - Terminates the program when the user chooses to exit.

# ❖ Pseudocode :

```
START

DECLARE voter array of struct { int voter_id; char name[40]; int voted } size 100
DECLARE candidates array of struct { int candidate_id; char name[40]; int votes }
with initial values [(1, "A", 0), (2, "B", 0), (3, "C", 0)]
DECLARE num_voter = 0
DECLARE i, voter_id, candidate_choice

FUNCTION main()
   DO
      PRINT menu options
      READ choice

      SWITCH choice
         CASE 1:
            CALL registerVoter()
         CASE 2:
            CALL vote()
         CASE 3:
            CALL voteCount()
         CASE 4:
            PRINT "exiting.."
            RETURN 0
         DEFAULT:
            PRINT "Invalid choice"
      END SWITCH
   WHILE true
END FUNCTION

FUNCTION registerVoter()
```

```
    PRINT "enter your name: "
    READ voter[num_voter].name
    SET voter[num_voter].voter_id TO num_voter + 1
    SET voter[num_voter].voted TO 0
    PRINT "registered successfully. voter id = " num_voter + 1
    INCREMENT num_voter
END FUNCTION

FUNCTION displayCandidate()
    PRINT "candidates: "
    FOR i FROM 0 TO 2
        PRINT i + 1, candidates[i].name
    END FOR
END FUNCTION

FUNCTION vote()
    PRINT "enter voter_id: "
    READ voter_id

    IF voter_id >= 1 AND voter_id <= 100 THEN
        IF voter[voter_id - 1].voter_id == voter_id THEN
            IF voter[voter_id - 1].voted == 0 THEN
                CALL displayCandidate()
                PRINT "enter your candidate choice (1 to 3): "
                READ candidate_choice

                IF candidate_choice < 1 OR candidate_choice > 3 THEN
                    PRINT "invalid choice"
                ELSE
                    SET voter[voter_id - 1].voted TO 1
                    INCREMENT candidates[candidate_choice - 1].votes
                    PRINT "voted successfully"
```

```
            END IF
        ELSE
            PRINT "you have already voted"
        END IF
    ELSE
        PRINT "Please register before voting"
    END IF
ELSE
    PRINT "invalid voter_id"
END IF
END FUNCTION

FUNCTION voteCount()
    PRINT "vote count"
    PRINT "candidate\tvotes"
    FOR i FROM 0 TO 2
        PRINT candidates[i].name, candidates[i].votes
    END FOR
END FUNCTION

END
```

# Testing and Results

## - Test cases:

To ensure the voting system works correctly, the following test cases can be designed:

- **Register Voter :**
  - **Test Case 1**: Register a new voter with a valid name.
    - **Input**: "Agent"
    - **Expected Output**: "Registered successfully. Voter ID = 1"
  - **Test Case 2**: Register another voter.
    - **Input**: "Arjun"
    - **Expected Output**: "Registered successfully. Voter ID = 2"
- **Vote :**
  - **Test Case 3**: Vote with a valid voter ID and candidateA choice.
    - **Input**: Voter ID = 1, Candidate Choice = 1
    - **Expected Output**: "Voted successfully"
  - **Test Case 4**: Attempt to vote with an invalid voter ID.
    - **LT**: Voter ID = 101, Candidate Choice = 1
    - **Expected Output**: "Invalid voter ID"
  - **Test Case 5**: Attempt to vote with a voter ID that has already voted.
    - **Input**: Voter ID = 1, Candidate Choice = 2
    - **Expected Output**: "You have already voted"
  - **Test Case 6**: Attempt to vote with an invalid candidate choice.
    - **Input**: Voter ID = 2, Candidate Choice = 4
    - **Expected Output**: "Invalid choice"
- **Vote Count :**
  - **Test Case 7**: Display vote count after multiple votes.
    - **Input**: None (just select "Count of votes" option)
    - **Expected Output**: "Vote count\nCandidate Votes\nA 1\nB 0\nC 0" (Assuming only one vote for Candidate A)
- **Exit:**

  **Test Case 8** : Exit Test Case: Exit from Voting System

  - **Input:** Select "4" from the menu to exit the program.
  - **Expected Output:** "Exiting.."

## - Output screenshots or results:

**Output:**

```
        Menu
 1. Register voter
 2. Vote
 3. count of votes
 4. Exit
 enter your choice :
```

- **Test Case 1**: Register a new voter with a valid name.

```
enter your choice : 1
enter your name: Agent

registered succesfully.  voter id =1
```

- **Test Case 2**: Registering another voter.

```
enter your choice : 1
enter your name: Arjun

registered succesfully.  voter id =2
```

- **Test Case 3**: Vote with a valid voter ID and candidate choice.

```
enter your choice : 2

enter voter_id: 1

candidates:
1. A
2. B
3. C
enter your candidate choice(1 to 3) : 1
voted succesfully
```

- **Test Case 4**: Attempt to vote with an invalid voter ID.

```
enter your choice : 2

enter voter_id: 101
invalid voter_id
```

- **Test Case 5**: Attempt to vote with a voter ID that has already voted.

```
enter your choice : 2

enter voter_id: 1
you have already voted
```

- **Test Case 6**: Attempt to vote with an invalid candidate choice.

```
enter your choice : 2

enter voter_id: 2

candidates:
1. A
2. B
3. C
enter your candidate choice(1 to 3) : 4
invalid choice
```

- **Test Case 7**: Display vote count after multiple votes.

```
enter your choice : 3
vote count
candidate       votes
 A               1
 B               0
 C               0
```

- **Test Case 8 :** Exit option in menu

```
enter your choice : 4
exiting..
```

# - Discussion of results:

The testing of the voting system confirms that it performs as intended across its main functions. Here are the key takeaways from the results:

- **Voter Registration**: The system effectively registers voters, assigns them unique IDs, and prevents any duplicate registrations.
- **Voting Process**: It validates voter IDs, ensuring only registered voters can vote. The system also safeguards against multiple votes from the same person and handles invalid candidate choices appropriately.
- **Vote Counting**: The system accurately tallies and displays votes for each candidate, providing a clear reflection of the election outcome.

Overall, these findings demonstrate that the voting system functions reliably, achieving its primary objectives effectively.

# Conclusion

## ➢ Summary of the Project :

This project developed a secure and efficient voting system using C programming. The system allows for:

1. **Voter Registration**: Ensuring each voter registers uniquely to prevent duplicate registrations.
2. **Voting Process**: Allowing registered voters to cast their votes securely and ensuring that each voter can vote only once.
3. **Vote Counting**: Accurately counting votes for each candidate and displaying the results.

The main goals were to improve security and integrity in the voting process, addressing issues such as double voting and registration inaccuracies.

## ➢ Future Enhancements :

To improve the system, we can consider the following future enhancements:

1. **User Authentication**: Implement stronger user authentication methods, like biometrics or two-factor authentication.
2. **Multilingual Support**: Support multiple languages to accommodate all voters in regions like Kerala.
3. **Encryption**: Use encryption to protect voter information and votes.

# Reference


- https://www.eci.gov.in/evm/
- https://www.sec.kerala.gov.in/

# Appendices

## - Source code :

```
#include <stdio.h>


void displayCandidate();

void registerVoter();

void vote();

void voteCount();


struct voting {

    int voter_id;

    char name[40];

    int voted;

    }voter[100];


struct candidate {

    int candidate_id;

    char name[40];

    int votes;

}candidates[10]={{1,"A",0},{2,"B",0},{3,"C",0}};


int num_voter = 0,i,voter_id,candidate_choice;
```

```
int main(){
    int choice;
    do {
        printf("\n\t Menu \n");
        printf("1. Register voter \n");
        printf("2. Vote \n");
        printf("3. count of votes \n");
        printf("4. Exit \n");
        printf("enter your choice : ");
        scanf("%d",&choice);

        switch(choice){
            case 1:
                registerVoter();
                break;
            case 2:

                vote();
                break;
            case 3:
                voteCount();
                break;
            case 4:
                printf("exiting..");
```

```
            return 0;


        default:
            printf("Invalid choice");
      }
   }while(1);
}

void registerVoter(){
   printf("enter your name: ");
   scanf(" %[^\n]",voter[num_voter].name);
   voter[num_voter].voter_id = num_voter+1;
   voter[num_voter].voted = 0;
   printf("\nregistered succesfully.  voter id =%d\n",++num_voter);
}

void displayCandidate(){
   printf("\ncandidates: \n");
   for(i=0;i<3;i++){
      printf("%d. %s\n",i+1,candidates[i].name);
   }
}

void vote(){
```

```
printf("\nenter voter_id: ");
scanf("%d",&voter_id);
if(voter_id>=1 && voter_id<=100){
    if(voter[voter_id-1].voter_id==voter_id){
        if(voter[voter_id-1].voted==0){
            displayCandidate();
            printf("enter your candidate choice(1 to 3) : ");
            scanf("%d",&candidate_choice);
            if(candidate_choice<1 ||candidate_choice>3){
                printf("invalid choice");
            }else{
                voter[voter_id-1].voted = 1;
                candidates[candidate_choice-1].votes++;
                printf("voted succesfully\n");
            }
        }else{
            printf("you have already voted");
        }
    }else{
        printf("Please register before voting\n");
    }
}else{
    printf("invalid voter_id");
}
```

```
}


void voteCount(){
    printf("vote count\n");
    printf("candidate\tvotes\n");
    for (int i = 0; i < 3; i++)
    {
        printf(" %s\t\t %d\n",candidates[i].name,candidates[i].votes);
    }

}
```