# VOTING PROCESS

Presented by

R.Ramyasree(24KB1A3089)

G.Harshitha(24KB1A3030)

J.Neha(24kB1A3037)

# ABSTRACT:


- This project is a basic voting system developed in C using the queue data structure.
- Voters are added to a queue and allowed to vote in the order they arrive,just like in a real polling station.
- Each voter has a name and must be verified before voting.
- Verified voters can choose one of the available candidates, and there are counted.
- The system also keeps track of who voted for each candidate and displays the final result with vote counts.
- This project helps to understand how queues work in real-world situations using Simple C programming.

## System requirements:

- C programming language.
- Basic knowledge of arrays and structures.
- Console-based interface for demonstration.

## System design overview:

- Voters register and are added to the queue.
- Each voter votes when their turn arrives(dequeued).
- Voters are recorded and tallied-Result displayed after all voters have voted.

# ***HEADER FILE:***

- #include<stdio.h> : Includes the standard input/output library which provides functions for input and output operations.
- #include<stdlib.h> : It provides functions for memory allocation, process control, conversions and other utilities.
- #include<string.h> : Includes standard library functions which provides functions for C strings such as strcpy( ),strlen( ) and strcmp( ).

# ***DEFINING CONSTANT:***

1. #define MAX_CANDIDATES:

- Sets the maximum number of candidates in the voting system to 3.
- Anywhere MAX_CANDIDATES appears in the code, it will be replaced with 3 by the preprocessor.

2. #define MAX_NAME_LENGTH 50:

- Sets the maximum length for a voter's name to 50 characters (including the null terminator).
- Used to define the size of character arrays that store names.

3. #define RESULT_FILE "voting_results.txt”:

- Defines the name of the file where voting results will be saved as "voting_results.txt".
- Used in file operations to refer to the results file.

4.#define MAX_VOTERS :

- Sets the maximum number of voters allowed in the system to 3.
- Used to define the size of arrays and limits for voter-related operations.

```
typedef struct Voter {
                    char name[MAX_NAME_LENGTH];
                    int id;    int verified;
                    int votedFor;
                    struct Voter* next;
 } Voter;
   typedef struct Queue {
            Voter* front;
           Voter* rear;
} Queue;
```

1. Structure Definition
2. Fields Explained
3. Typedef Usage

# LOGIC CODE:

```
void startVoting(Queue* q) {
    Voter* voter;
    while ((voter = dequeue(q)) != NULL) {
        printf("\nVoter Name: %s\n", voter->name);

        if (!voter->verified) {
            printf("You are not verified to vote.\n");
            free(voter);
            continue;
        }

        printf("Select your candidate:\n");
        for (int i = 0; i < MAX_CANDIDATES; i++) {
            printf("%d. %s\n", i + 1, candidateNames[i]);
        }
```

```
        int choice;
        printf("Enter your choice (1-%d): ", MAX_CANDIDATES);
        scanf("%d", &choice);

        if (choice >= 1 && choice <= MAX_CANDIDATES) {
            candidateVotes[choice - 1]++;
            strcpy(votersForCandidates[choice - 1][votersCount[choice - 1]+
voter->name);
            printf("Thank you for voting!\n");
        } else {
            printf("Invalid choice.\n");
        }
        free(voter);
    }
}
```

# DISPLAY RESULTS:

```
void displayResults() {
    printf("\n--- Voting Results ---\n");
    int maxVotes = 0;
    int winningCandidate = -1;

    for (int i = 0; i < MAX_CANDIDATES; i++) {
        printf("%s: %d votes\n", candidateNames[i],
candidateVotes[i]);
        if (candidateVotes[i] > maxVotes) {
            maxVotes = candidateVotes[i];
            winningCandidate = i;
        }
    }

    if (winningCandidate != -1) {
        printf("\nWinner: %s with %d votes\n",
candidateNames[winningCandidate], maxVotes);
    } else {
        printf("No winner yet.\n");
    }
}
```

# SAMPLE OUTPUT:

```
--- Voting System ---
1. Start Voting
2. Display Results
3. Admin Menu
4. Exit

--- Voting Started ---

Voter Name: Neha
Select your candidate:
1. Candidate A
2. Candidate B
3. Candidate C
Enter your choice (1-3): Thank you for votir
```

```
Voter Name: Ramya
Select your candidate:
1. Candidate A
2. Candidate B
3. Candidate C
Enter your choice (1-3): Thank you for voting!

Voter Name: Harshi
Select your candidate:
1. Candidate A
2. Candidate B
3. Candidate C
Enter your choice (1-3): Thank you for voting!
```

```
--- Voting Results ---
Candidate A: 1 votes
Candidate B: 2 votes
Candidate C: 0 votes

Winner: Candidate B with 2 votes
```

- **ADVANTAGES OF QUEUE BASED ON VOTING SYSTEMS:**
- Simple, fair, and prevents manipulation.
- Easy to implement and debug.
- Scalable for larger scales.

**LIMITATIONS AND IMPROVEMENTS:**

- Limited by queue size(array implementation).
- No real-time authentication in console version.
- Can be extended with linked list for dynamic size.

# CONCLUSION:

- Queue ensures fair,sequential voting.
- C implementation demonstrates core data structure concepts.
- Foundation for more advanced, secure voting systems.

# FUTURE ENHANCEMENTS:

1. Unique voter identification:

- Assign each voter a unique ID to ensure **one vote per person.**
- Prevents impersonalisation and duplicate voting.

2. Biometric or OTP Verification:

- Simulate biometric or OTP-based verification to verify voter identity before voting.

# *THANK YOU*
# *ANY SUGGESTIONS*