PPROJECT TITLE :

Online Voting System

COURSE TEACHER :

ANIS AHMED

Md. SAJJAD HOSSAIN

Contributors :

Sayod Ittefar Hasan Jisan (2203105)
Abdur Rahman (2203106)
K.M. Safkut Reza (2203107)

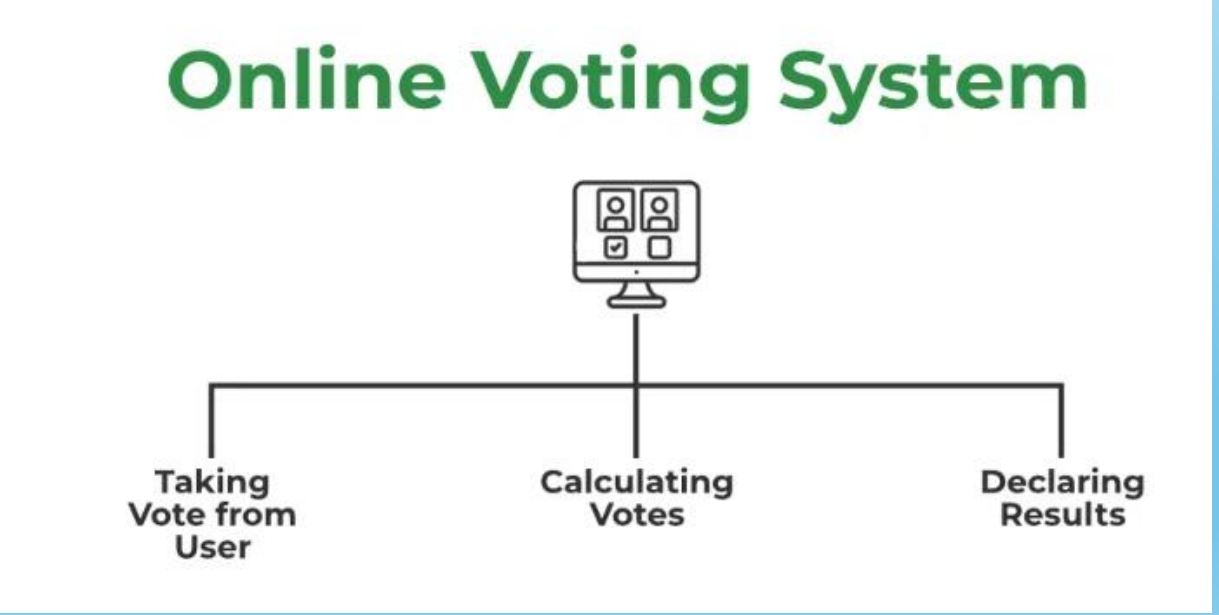
❖ What is Electronic Voting Machine ?

Voting in elections is one of the most fundamental organs of any Democracy and in the modern world, voting is done through digital machines. The device we use for online voting system is called EVM. An Electronic Voting Machine (EVM) is a device used to cast and count votes in elections. Instead of using paper ballots, voters make their selections on the machine, which then records and stores the votes electronically.

❑ Features of Online Voting System in C

The online voting system offers the following functions:

- Taking a Vote from the User
- Storing Different Votes
- Calculating Votes
- Declaring Results



➢ C Functions Used In This Code :

•**fillCandidate(int cNum)**: Collects and stores information about a candidate, including selecting a unique symbol and entering the candidate’s name.

•**displayAllCandidates()**: Displays all candidates' names and their chosen symbols in a formatted manner.

•**getVotes(int voterCount)**: Collects votes from voters, validates their choices, and updates the vote count.

•**getResults()**: Determines the outcome of the election by finding the candidate(s) with the most votes and displaying the result.

Includes and Definitions

```c
#include <stdio.h>
#include <string.h>
#define MAX_C 11
```

- stdio.h and string.h libraries are included for input/output operations and string manipulation.
- MAX_C is defined as 11 to set the maximum number of candidates (though the actual limit is 10).

Candidate Structure and Global Variables

```c
typedef struct Candidate {
    char name[50];
    int votes;
    char symbol;
} Candidate;

Candidate allCandidates[MAX_C];

int candidateCount = 0;

char symbols[10]
    = { '!', '@', '#', '$', '%', '^', '&', '*', '~', '+' };

int symbolTaken[11];
```

- Candidate is a structure holding the candidate's name, the number of votes, and a unique symbol.
- allCandidates is an array to store the candidates.
- candidateCount tracks the number of candidates.
- symbols contains the available symbols for candidates.
- symbolTaken is used to track which symbols have been assigned.

Function Declarations

```
void fillCandidate(int);
void displayAllCandidates();
void getVotes(int);
void getResults();
```

Main Function

```
int main()
{
    for (int i = 0; i < 11; i++) {
        symbolTaken[i] = 0;
    }
printf("Enter the number of candidates: ");
    scanf("%d", &candidateCount);
    if (candidateCount >= MAX_C) {
        printf("Number of candidates cannot be greater "
            "than 10.\n Terminating the program\n\n");
        return 0;
    }

    for (int i = 0; i < candidateCount; i++) {
        fillCandidate(i);
    }

    int numVoters;
    printf("Enter the number of voters: ");
    scanf("%d", &numVoters);

    for (int i = 0; i < numVoters; i++) {
        getVotes(i);
    }

    getResults();

    return 0;
}
```

- Initializes symbolTaken to all zeros.
- Reads the number of candidates and validates it.
- Calls fillCandidate to gather details for each candidate.
- Reads the number of voters and processes their votes.
- Calls getResults to determine and display the result.

fillCandidate Function

```
void fillCandidate(int cNum)
{
    printf("Available Symbols: \n");
    for (int j = 0; j < 10; j++) {
        if (symbolTaken[j] == 1)
            continue;
        printf("%d %c\n", j + 1, symbols[j]);
    }

    int num = 0;

    printf("\nEnter the symbol number of candidate %d: ",
        cNum + 1);
    scanf("%d", &num);

    if (num <= 0 || num > 10 || symbolTaken[num - 1] == 1) {
        printf("This Symbol is not available. Please "
            "choose from the available symbols\n");
        num = 0;
        fillCandidate(cNum);
    }
    else {
        symbolTaken[num - 1] = 1;
        allCandidates[cNum].symbol = symbols[num - 1];
        printf("Enter the name of candidate %d: ",
            cNum + 1);
        scanf("%s", allCandidates[cNum].name);

        allCandidates[cNum].votes = 0;
    }
}
```

- Displays available symbols.
- Asks for and validates the symbol choice for the candidate.
- Stores the symbol and candidate's name, initializing votes to 0.

displayAllCandidates Function

```
void displayAllCandidates()
{
    if (!allCandidates || !candidateCount) {
        perror("Invalid Candidate Array\n");
        return;
    }

    for (int j = 0; j < candidateCount; j++) {
        printf("%s\t\t", allCandidates[j].name);
    }
    printf("\n");
    for (int j = 0; j < candidateCount; j++) {
        printf("%3c\t\t\t", allCandidates[j].symbol);
    }
    printf("\n");
}
```

- Displays all candidates' names and their symbols.

getVotes Function

```
void getVotes(int voterCount)
{
    displayAllCandidates();
    printf("Voter %d, please enter your choice (1-%d): ",
        voterCount + 1, candidateCount);
    int choice;
    scanf("%d", &choice);

    if (choice >= 1 && choice <= candidateCount) {
        allCandidates[choice - 1].votes++;
    }
    else {
        printf("Invalid choice! Please vote again.\n");
        getVotes(voterCount);
    }
}
```

- Displays candidates and prompts the voter to enter their choice.
- Increments the vote count for the chosen candidate.
- Recursively asks for a valid vote if an invalid choice is entered.

getResults Function

```
void getResults()
{
    int maxVotes = 0;
    int winnerIndex = -1;
    int winnerFrequency = 0;
    for (int i = 0; i < candidateCount; i++) {
        if (allCandidates[i].votes > maxVotes) {
            maxVotes = allCandidates[i].votes;
            winnerIndex = i;
        }
    }

    for (int i = 0; i < candidateCount; i++) {
        if (allCandidates[i].votes == maxVotes) {
            winnerFrequency++;
        }
    }

    printf("\n-----RESULT-----\n");

    if (winnerFrequency > 1) {
        printf("No candidate has majority votes\n");
    }
    else if (winnerIndex != -1) {
        printf("The winner is: %s\nCandidate Symbol: "
            "%c\nwith %d votes!\n",
            allCandidates[winnerIndex].name,
            allCandidates[winnerIndex].symbol, maxVotes);
    }
    else {
        printf("No winner\n");
    }
}
```

- Determines the candidate with the most votes.
- Checks if there is a tie (more than one candidate with the maximum votes).
- Displays the result, including the winner's name, symbol, and vote count or states that no clear winner exists.

Result:

```
Enter the number of candidates: 2
Available Symbols:
1 !
2 @
3 #
4 $
5 %
6 ^
7 &
8 *
9 ~
10 +

Enter the symbol number of candidate 1: 3
Enter the name of candidate 1: Rahim
Available Symbols:
1 !
2 @
4 $
5 %
6 ^
7 &
8 *
9 ~
10 +

Enter the symbol number of candidate 2: 5
Enter the name of candidate 2: Karim
Enter the number of voters: 10
```

```
Rahim           Karim
  #                     %
Voter 1, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 2, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 3, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 4, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 5, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 6, please enter your choice (1-2): 2
Rahim           Karim
  #                     %
Voter 7, please enter your choice (1-2): 2
Rahim           Karim
  #                     %
Voter 8, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 9, please enter your choice (1-2): 1
Rahim           Karim
  #                     %
Voter 10, please enter your choice (1-2): 2

-----RESULT-----
The winner is: Rahim
Candidate Symbol: #
with 7 votes!

Process returned 0 (0x0)   execution time : 58.597 s
Press any key to continue.
```

Thank You for Attention