Problem-Solving Case Study – City Council Election

- Problem: In a city council election, citizens in four voting precincts have cast their ballots for four candidates. We want to know how many votes each candidate received in each precinct, how many total votes each candidate received, and how many total votes were cast in each precinct.
- Input: An arbitrary number of votes in a file `voteFile`, with each vote represented as a pair of numbers: a precinct number and a candidate number. The candidate names from keyboard.

Problem-Solving Case Study – City Council Election

- Output: A tabular report showing how many votes each candidate received in each precinct, the total number of votes for each candidate, and the total number of votes in each precinct, all written to `reportFile`.
- Discussion: If I were doing this by hand, I would probably write down the candidate names as the row labels of a table and use the precinct numbers as the column labels and put a tally mark in the appropriate place as I read each vote.

Problem-Solving Case Study – City Council Election

- More Discussion: A table is easily represented by a two-dimensional array. We can save the names of the candidates in a one-dimensional array. The book example uses precincts for row headings and candidate numbers for column headings.
- Data Structures: A two-dimensional array names `votes`, where the rows represent precincts and the columns represent candidates. A one-dimensional array of strings containing the names of the candidates.

Main

```
Open voteFile for input (and verify success)
Open reportFile for output (and verify success)
Get candidate names
Set votes array to 0
Read precinct, candidate from voteFile
WHILE NOT EOF on voteFile
    Increment votes[precinct-1][candidate-1]
    Read precinct, candidate from voteFile
Write report to reportFile
Write totals per candidate to reportFile
Write totals per precinct to reportFile
```

Write Totals per Precinct (In: `votes`; Inout: `reportFile`)

```
FOR each precinct
    Set total = 0

    // Compute row sum
    FOR each candidate
        Add votes[precinct][candidate] to
            total
    Write "Total votes for precinct", precinct+1,
    ':', total to reportFile
```

Implementation

```
//****************************************************************
// Election program
// This program reads votes represented by precinct number and
// ballot position from a data file, calculates the sums per
// precinct and per candidate, and writes all totals to an
// output file
//****************************************************************
#include …

using namespace std;

const int NUM_PRECINCTS = 4;
const int NUM_CANDIDATES = 4;

typedef int VoteArray[NUM_PRECINCTS][NUM_CANDIDATES];
                                 // 2-dimensional array type
                                 //   for votes
```

Implementation

```
void GetNames( string[] );
void OpenForInput( ifstream& );
void OpenForOutput( ofstream& );
void WritePerCandidate(const VoteArray, const string[],
                        ofstream& );
void WritePerPrecinct( const VoteArray, ofstream& );
void WriteReport( const VoteArray, const string[], ofstream& );
void ZeroVotes( VoteArray );

int main()
{
    string    name[NUM_CANDIDATES];  // Array of candidate names
    VoteArray votes;        // Totals for precincts vs. candidates
    int       candidate;   // Candidate number input from voteFile
    int       precinct;    // Precinct number input from voteFile
    ifstream  voteFile;     // Input file of precincts, candidates
    ofstream  reportFile;    // Output file receiving summaries
```

Implementation

```
    OpenForInput(voteFile);
    if ( !voteFile )
        return 1;
    OpenForOutput(reportFile);
    if ( !reportFile )
        return 1;

    GetNames(name);
    ZeroVotes(votes);

    // Read and tally votes
    voteFile >> precinct >> candidate;
    while (voteFile)
    {
        votes[precinct-1][candidate-1]++;
        voteFile >> precinct >> candidate;
    }
```

Implementation

```
    // Write results to report file

    WriteReport(votes, name, reportFile);
    WritePerCandidate(votes, name, reportFile);
    WritePerPrecinct(votes, reportFile);

    return 0;
}
```

Implementation

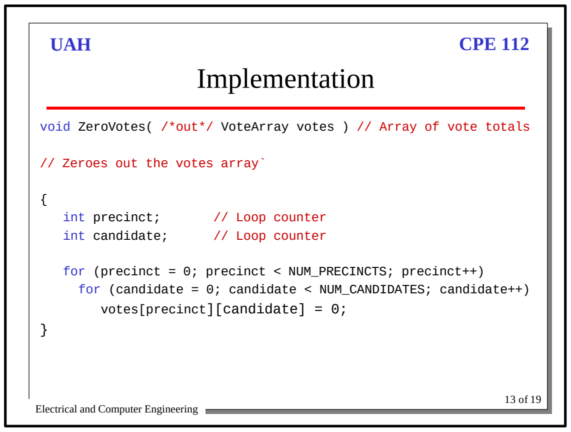
```
void OpenForInput( /*inout*/ ifstream& someFile )  // File to be
                                                     // opened
// Prompts the user for the name of an input file
// and attempts to open the file

{
    string fileName;    // User-specified file name

    cout << "Input file name: ";
    cin >> fileName;

    someFile.open(fileName.c_str());
    if ( !someFile )
        cout << "** Can't open " << fileName << " **" << endl;
}
```

Implementation

```
void OpenForOutput( /*inout*/ ofstream& someFile )  // File to be
                                                      // opened
{
    string fileName;    // User-specified file name

    cout << "Output file name: ";
    cin >> fileName;

    someFile.open(fileName.c_str());
    if ( !someFile )
        cout << "** Can't open " << fileName << " **" << endl;
}
```

Implementation

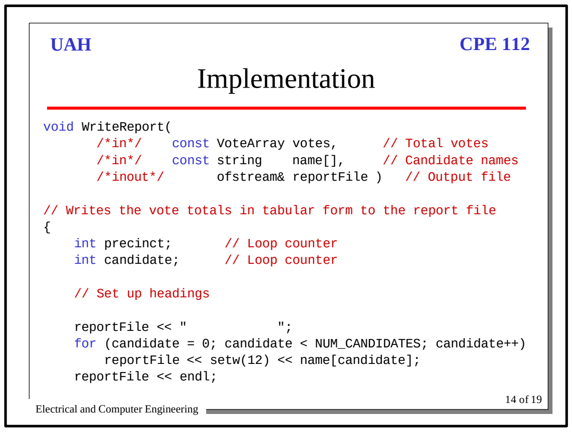
```
void GetNames( /*out*/ string name[] )    // Array of candidate
                                          //   names
// Reads the candidate names from standard input
{
    string inputStr;      // An input string
    int    candidate;     // Loop counter

    cout << "Enter the names of the candidates, one per line,"
         << endl << "in the order they appear on the ballot."
         << endl;

    for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
    {
        cin >> inputStr;
        name[candidate] = inputStr.substr(0, 10);
    }
}
```


UAH CPE 112

Implementation

```
void ZeroVotes( /*out*/ VoteArray votes ) // Array of vote totals

// Zeroes out the votes array`

{
   int precinct;       // Loop counter
   int candidate;      // Loop counter

   for (precinct = 0; precinct < NUM_PRECINCTS; precinct++)
     for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
        votes[precinct][candidate] = 0;
}
```

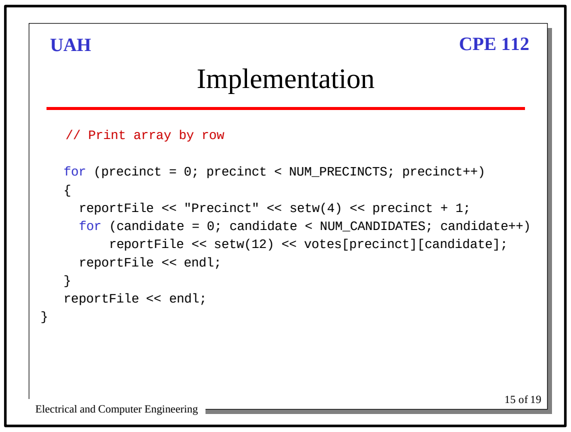
Electrical and Computer Engineering 13 of 19


UAH CPE 112

Implementation

```
void WriteReport(
       /*in*/    const VoteArray votes,      // Total votes
       /*in*/    const string    name[],     // Candidate names
       /*inout*/       ofstream& reportFile )   // Output file

// Writes the vote totals in tabular form to the report file
{
    int precinct;       // Loop counter
    int candidate;      // Loop counter

    // Set up headings

    reportFile << "            ";
    for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
        reportFile << setw(12) << name[candidate];
    reportFile << endl;
```

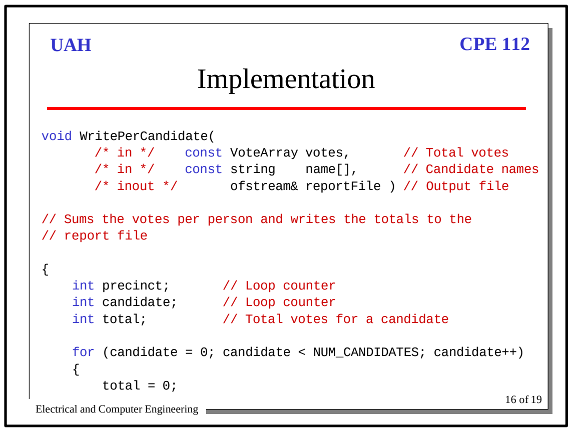
Electrical and Computer Engineering 14 of 19


UAH CPE 112

Implementation

```
   // Print array by row

   for (precinct = 0; precinct < NUM_PRECINCTS; precinct++)
   {
     reportFile << "Precinct" << setw(4) << precinct + 1;
     for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
         reportFile << setw(12) << votes[precinct][candidate];
     reportFile << endl;
   }
   reportFile << endl;
}
```

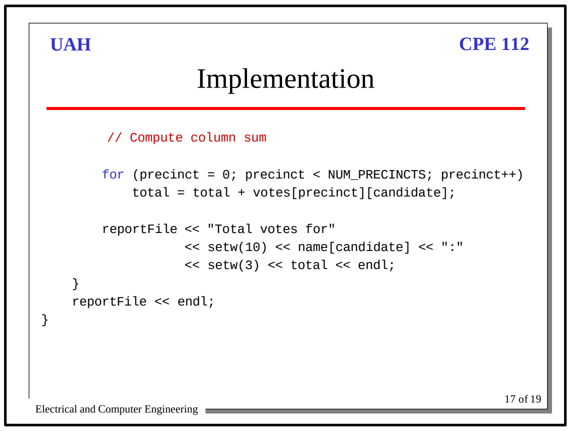
Electrical and Computer Engineering 15 of 19


UAH CPE 112

Implementation

```
void WritePerCandidate(
       /* in */    const VoteArray votes,       // Total votes
       /* in */    const string    name[],      // Candidate names
       /* inout */       ofstream& reportFile ) // Output file

// Sums the votes per person and writes the totals to the
// report file

{
    int precinct;       // Loop counter
    int candidate;      // Loop counter
    int total;          // Total votes for a candidate

    for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
    {
        total = 0;
```

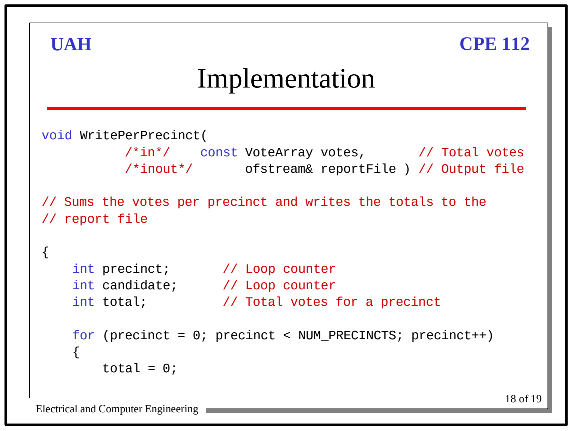
Electrical and Computer Engineering 16 of 19


UAH CPE 112

Implementation

```
        // Compute column sum

        for (precinct = 0; precinct < NUM_PRECINCTS; precinct++)
            total = total + votes[precinct][candidate];

        reportFile << "Total votes for"
                   << setw(10) << name[candidate] << ":"
                   << setw(3) << total << endl;
    }
    reportFile << endl;
}
```

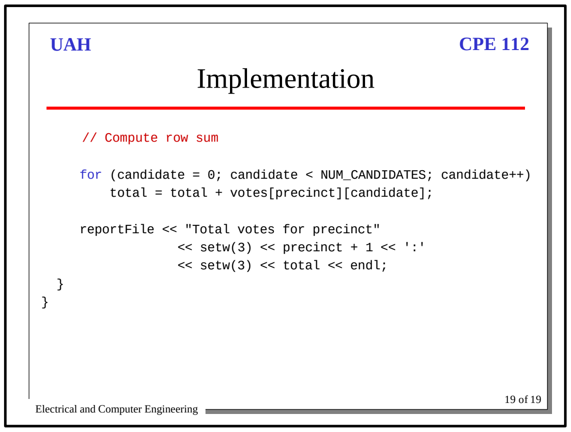
Electrical and Computer Engineering 17 of 19


UAH CPE 112

Implementation

```
void WritePerPrecinct(
           /*in*/    const VoteArray votes,       // Total votes
           /*inout*/       ofstream& reportFile ) // Output file

// Sums the votes per precinct and writes the totals to the
// report file

{
    int precinct;       // Loop counter
    int candidate;      // Loop counter
    int total;          // Total votes for a precinct

    for (precinct = 0; precinct < NUM_PRECINCTS; precinct++)
    {
        total = 0;
```

Electrical and Computer Engineering 18 of 19



Implementation

```
    // Compute row sum

    for (candidate = 0; candidate < NUM_CANDIDATES; candidate++)
        total = total + votes[precinct][candidate];

    reportFile << "Total votes for precinct"
                 << setw(3) << precinct + 1 << ':'
                 << setw(3) << total << endl;
  }
}
```