계산과학 이론 및 실습2

comparing the plurality representation system & the proportional representation system in political election

made by Jeongwoo Kim

Background – Korean assembly election results

		Democratic Party	People Power Party	Justice Party
2016	Ratio of votes	25.5%	33.5%	7.2%
	Seats	123	122	6
2020	Ratio of votes	33.4	33.8%	9.7%
	Seats	180	103	6

Background

the plurality representation system

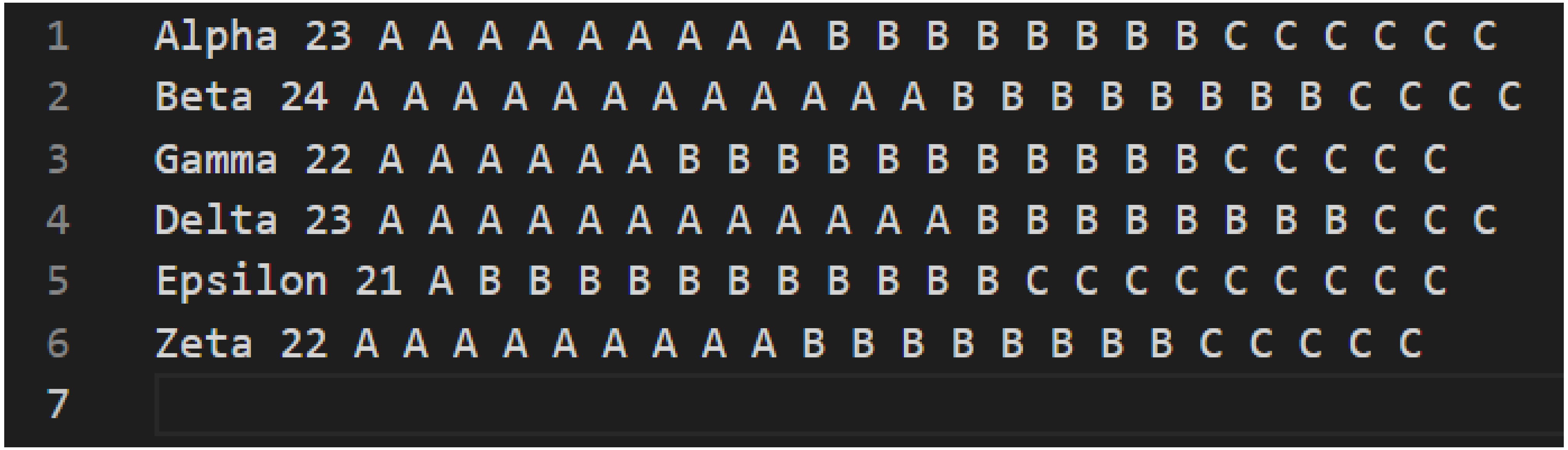
: The candidate with the most votes in each constituency is elected.
=> It is known that there is a large discrepancy between the percentage of votes and the number of seats won.

vs

the proportional representation system

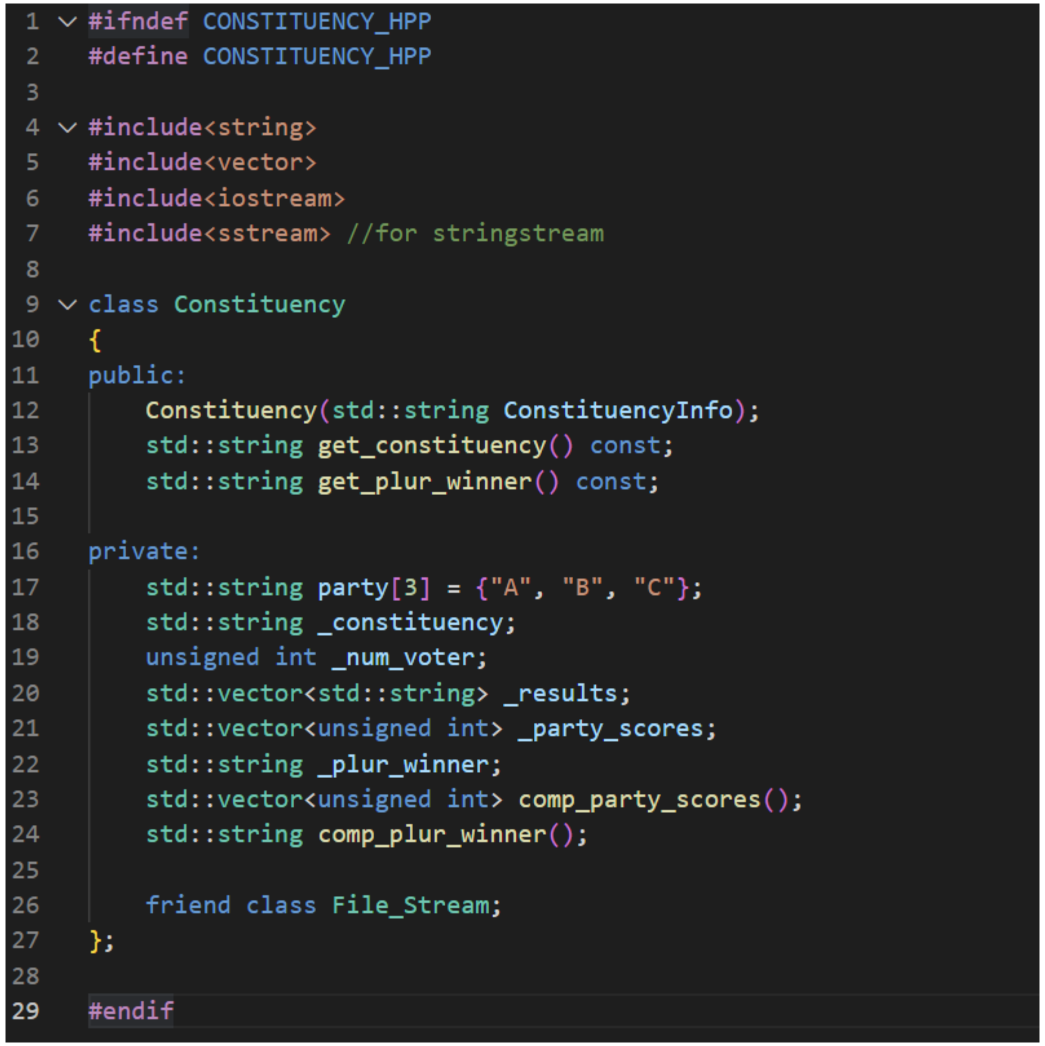
: The number of votes in all constituencies is combined, and seats are allocated to political parties according to the proportion.
=> It is known that small parties can win seats more easily.

Input file

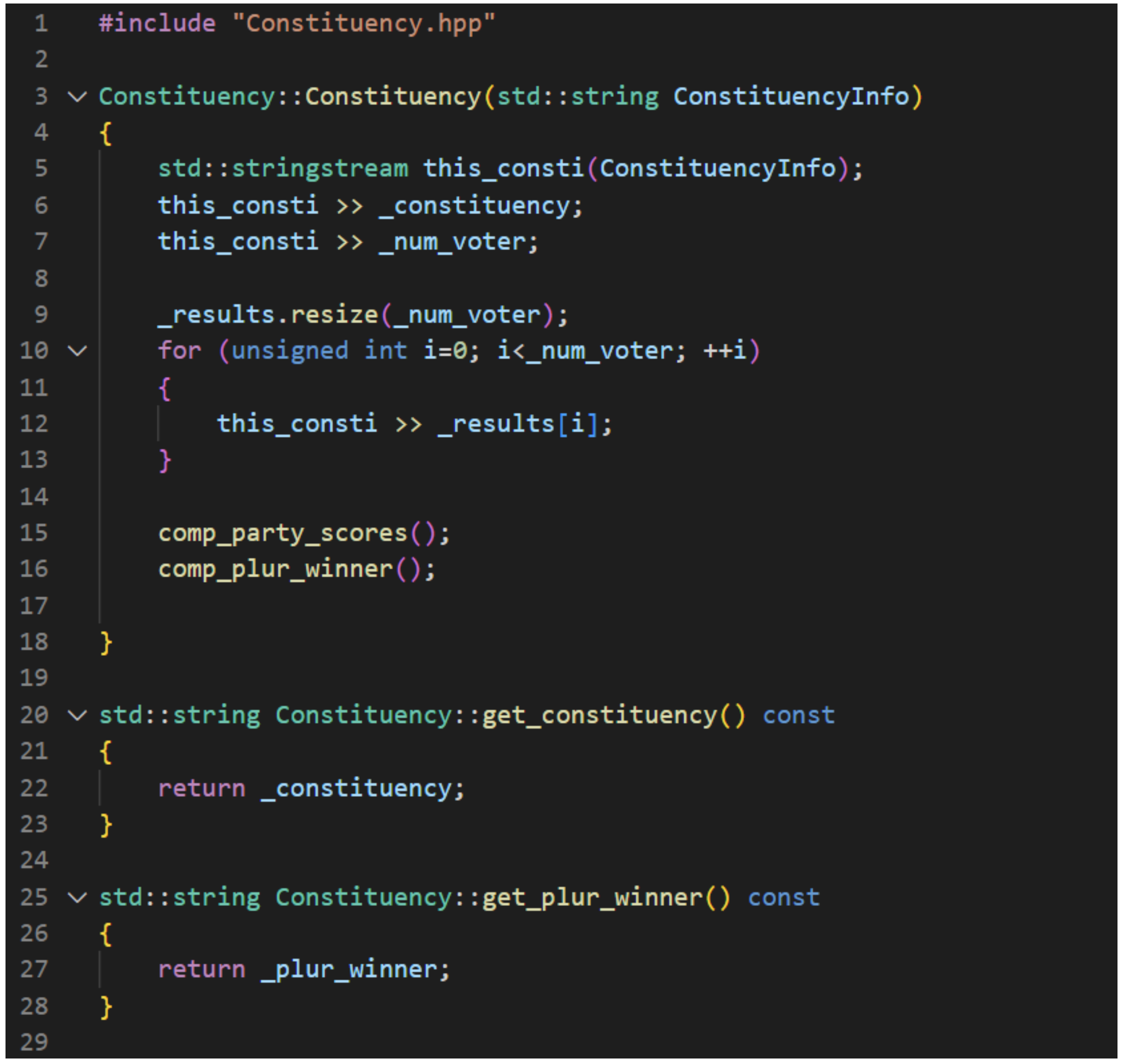


```
Alpha 23 A A A A A A A A A B B B B B B B B C C C C C C
Beta 24 A A A A A A A A A A A A B B B B B B B B C C C C
Gamma 22 A A A A A A B B B B B B B B B B B C C C C C
Delta 23 A A A A A A A A A A A A B B B B B B B B C C C
Epsilon 21 A B B B B B B B B B B B C C C C C C C C C
Zeta 22 A A A A A A A A A B B B B B B B B C C C C C

```

Program code: Constituency.hpp & .cpp

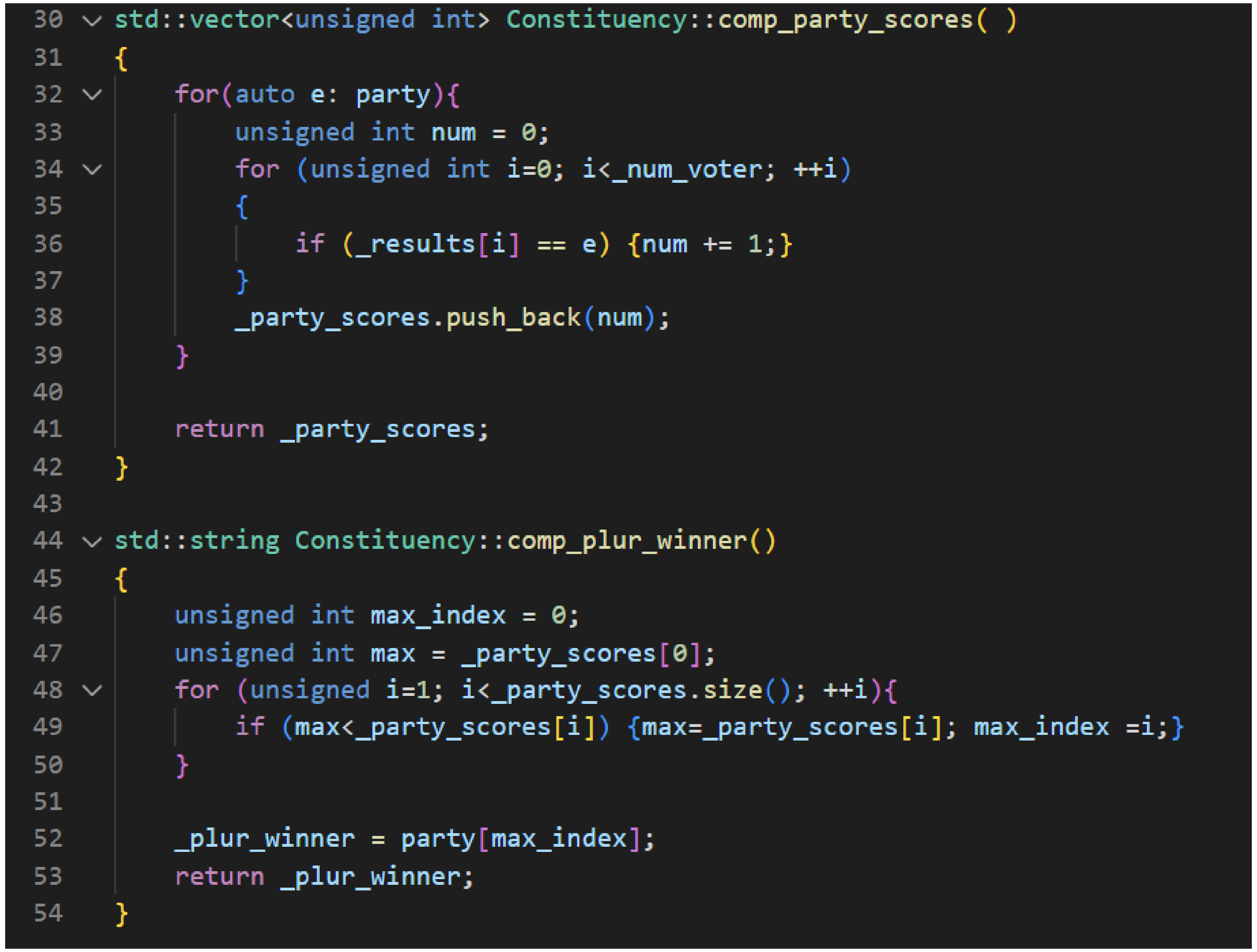


```cpp
#ifndef CONSTITUENCY_HPP
#define CONSTITUENCY_HPP

#include<string>
#include<vector>
#include<iostream>
#include<sstream> //for stringstream

class Constituency
{
public:
    Constituency(std::string ConstituencyInfo);
    std::string get_constituency() const;
    std::string get_plur_winner() const;

private:
    std::string party[3] = {"A", "B", "C"};
    std::string _constituency;
    unsigned int _num_voter;
    std::vector<std::string> _results;
    std::vector<unsigned int> _party_scores;
    std::string _plur_winner;
    std::vector<unsigned int> comp_party_scores();
    std::string comp_plur_winner();

    friend class File_Stream;
};

#endif
```



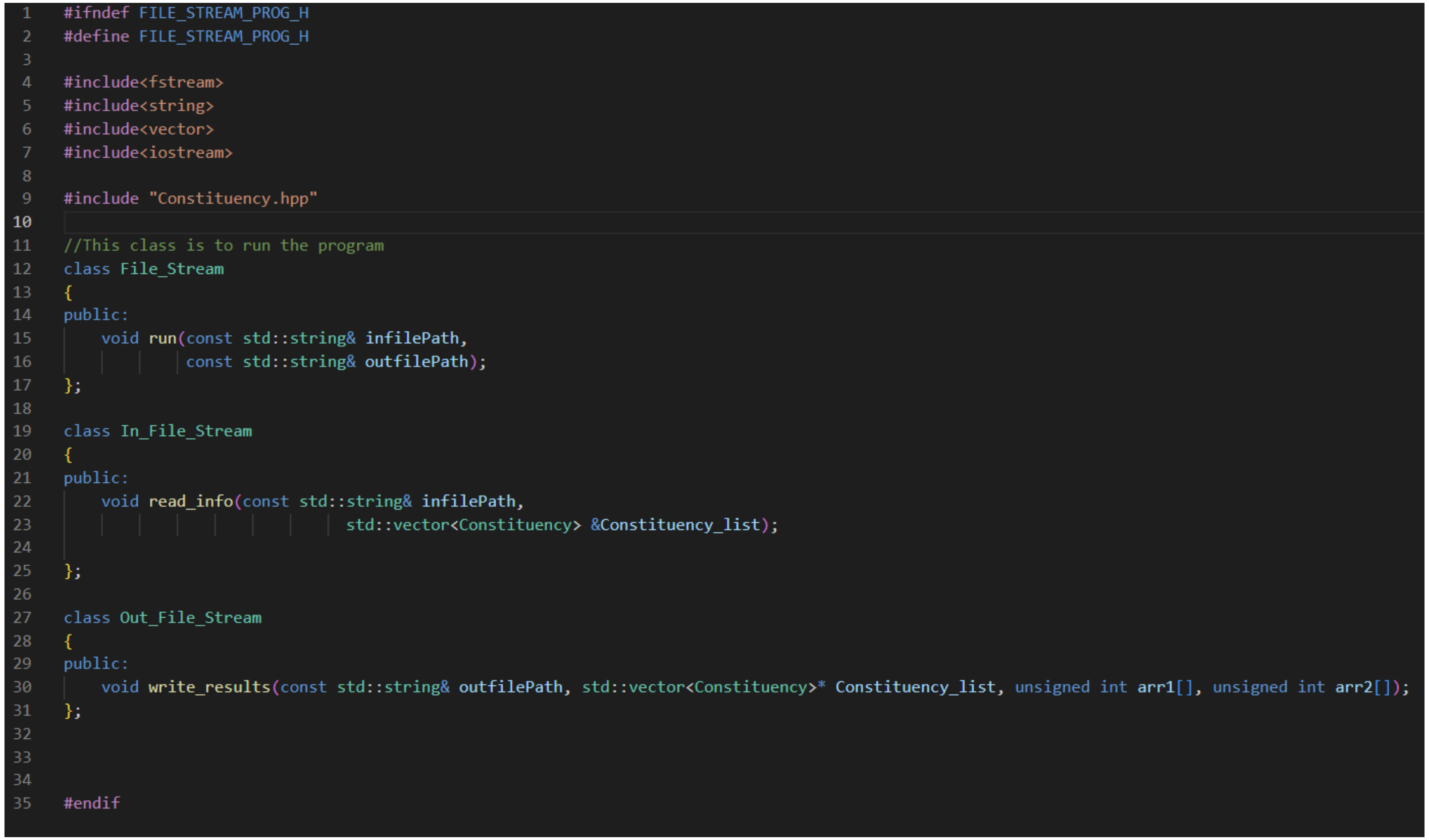
```cpp
#include "Constituency.hpp"

Constituency::Constituency(std::string ConstituencyInfo)
{
    std::stringstream this_consti(ConstituencyInfo);
    this_consti >> _constituency;
    this_consti >> _num_voter;

    _results.resize(_num_voter);
    for (unsigned int i=0; i<_num_voter; ++i)
    {
        this_consti >> _results[i];
    }

    comp_party_scores();
    comp_plur_winner();

}

std::string Constituency::get_constituency() const
{
    return _constituency;
}

std::string Constituency::get_plur_winner() const
{
    return _plur_winner;
}

```

Program code: Constituency.hpp & .cpp



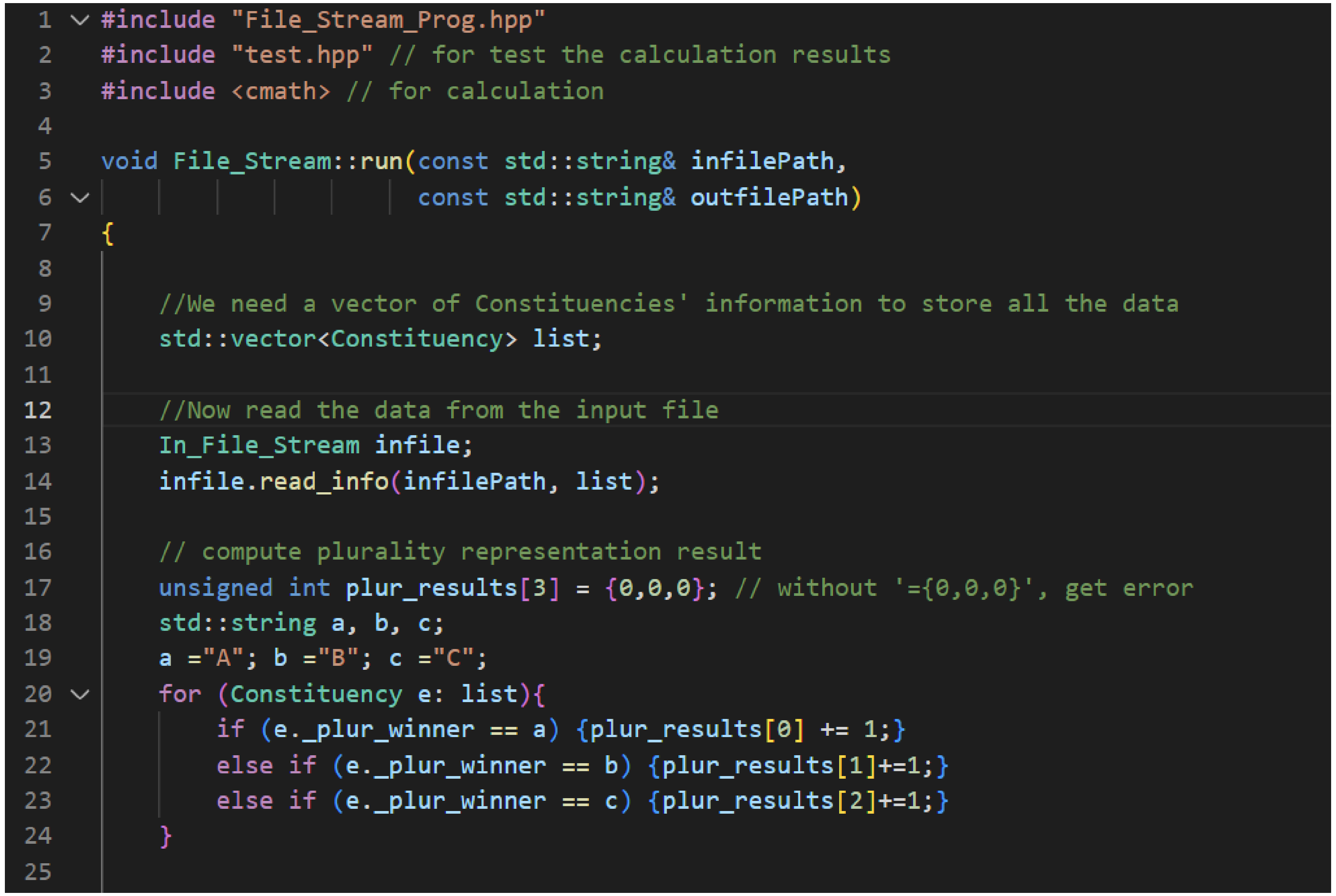
```
std::vector<unsigned int> Constituency::comp_party_scores( )
{
    for(auto e: party){
        unsigned int num = 0;
        for (unsigned int i=0; i<_num_voter; ++i)
        {
            if (_results[i] == e) {num += 1;}
        }
        _party_scores.push_back(num);
    }

    return _party_scores;
}

std::string Constituency::comp_plur_winner()
{
    unsigned int max_index = 0;
    unsigned int max = _party_scores[0];
    for (unsigned i=1; i<_party_scores.size(); ++i){
        if (max<_party_scores[i]) {max=_party_scores[i]; max_index =i;}
    }

    _plur_winner = party[max_index];
    return _plur_winner;
}
```

Program code: File_Stream_Prog.hpp & .cpp

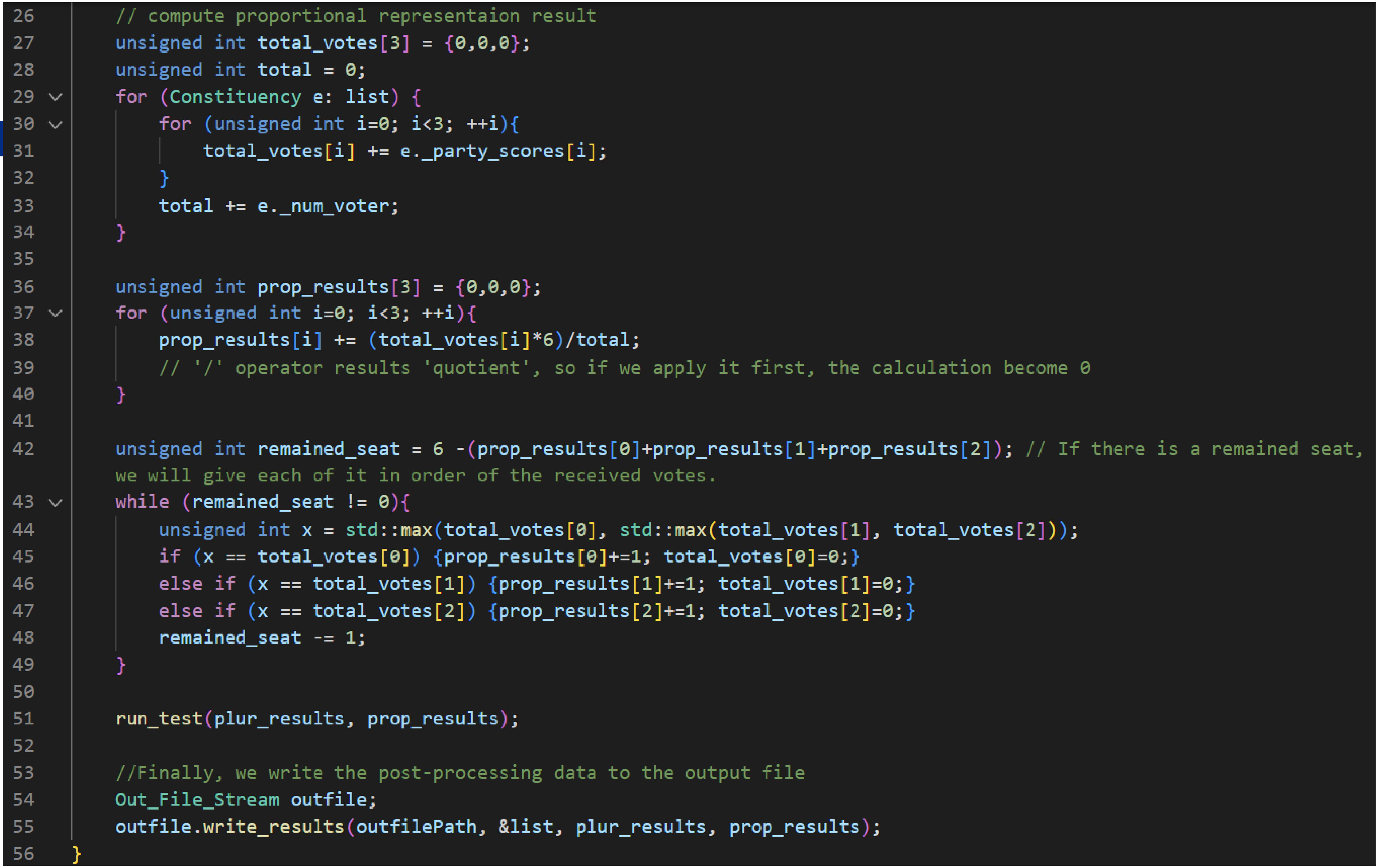


```
#ifndef FILE_STREAM_PROG_H
#define FILE_STREAM_PROG_H

#include<fstream>
#include<string>
#include<vector>
#include<iostream>

#include "Constituency.hpp"

//This class is to run the program
class File_Stream
{
public:
    void run(const std::string& infilePath,
             const std::string& outfilePath);
};

class In_File_Stream
{
public:
    void read_info(const std::string& infilePath,
                   std::vector<Constituency> &Constituency_list);

};

class Out_File_Stream
{
public:
    void write_results(const std::string& outfilePath, std::vector<Constituency>* Constituency_list, unsigned int arr1[], unsigned int arr2[]);
};



#endif
```

Program code: File_Stream_Prog.hpp & .cpp

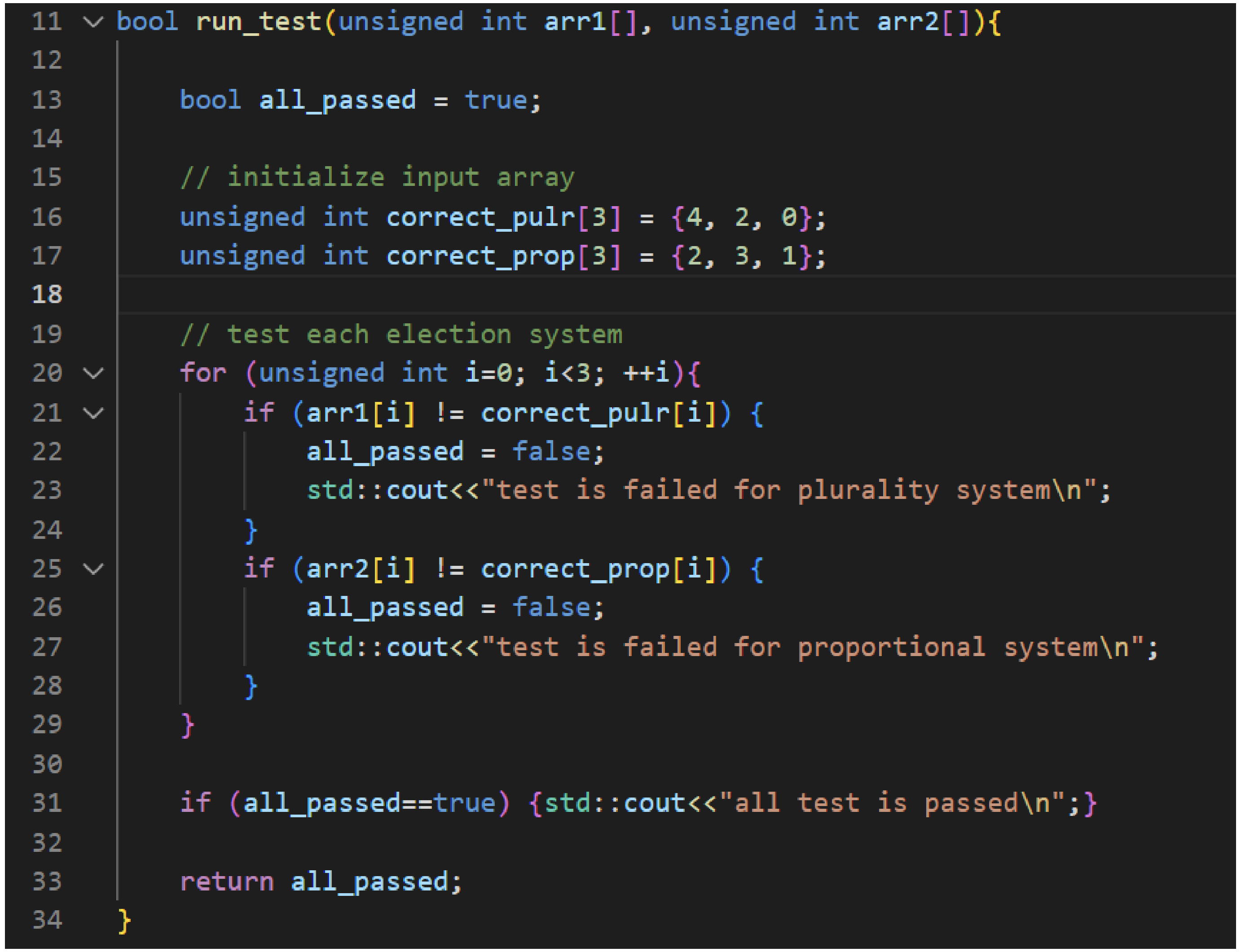


```cpp
#include "File_Stream_Prog.hpp"
#include "test.hpp" // for test the calculation results
#include <cmath> // for calculation

void File_Stream::run(const std::string& infilePath,
                      const std::string& outfilePath)
{

    //We need a vector of Constituencies' information to store all the data
    std::vector<Constituency> list;

    //Now read the data from the input file
    In_File_Stream infile;
    infile.read_info(infilePath, list);

    // compute plurality representation result
    unsigned int plur_results[3] = {0,0,0}; // without '={0,0,0}', get error
    std::string a, b, c;
    a ="A"; b ="B"; c ="C";
    for (Constituency e: list){
        if (e._plur_winner == a) {plur_results[0] += 1;}
        else if (e._plur_winner == b) {plur_results[1]+=1;}
        else if (e._plur_winner == c) {plur_results[2]+=1;}
    }

```



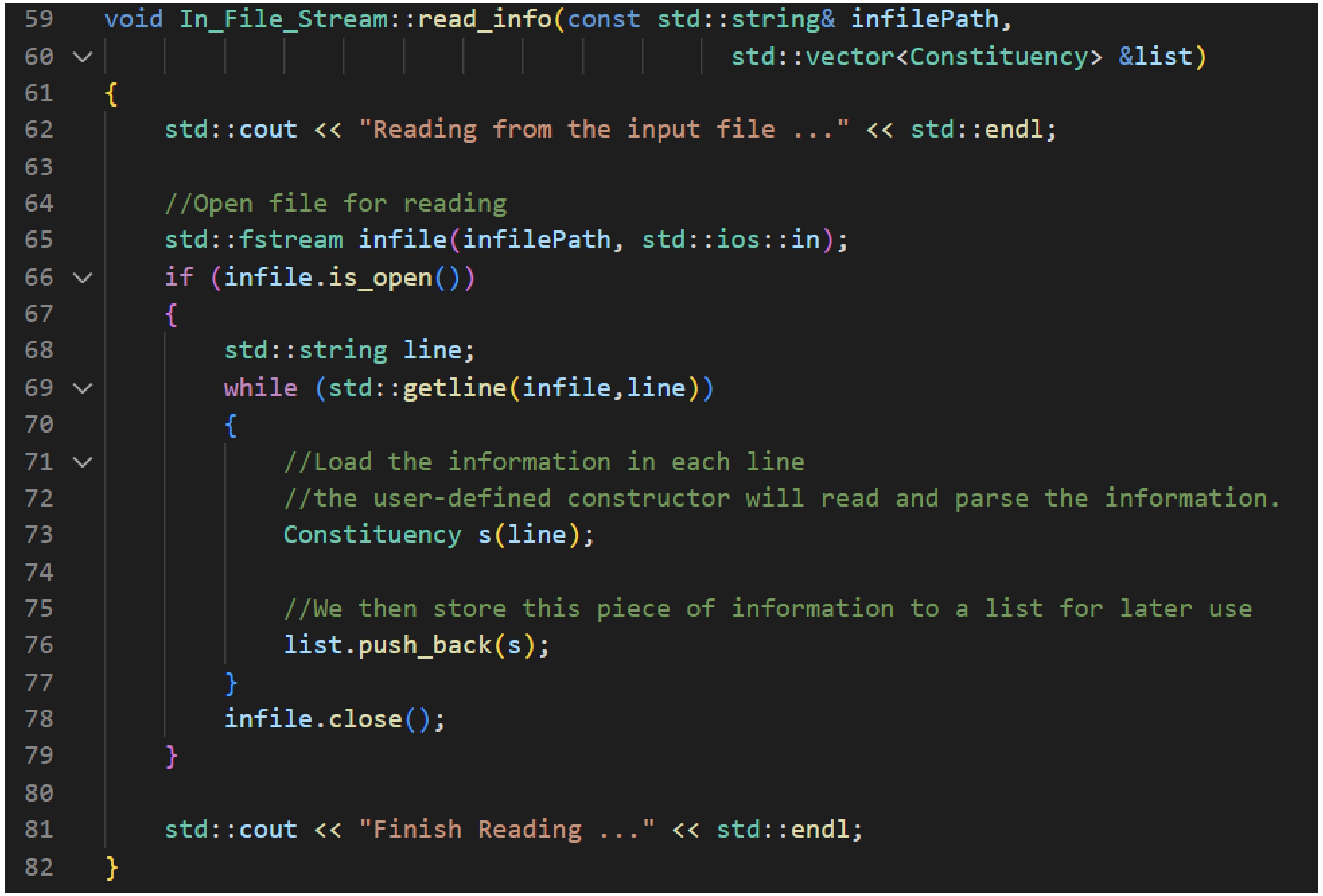
```
    // compute proportional representaion result
    unsigned int total_votes[3] = {0,0,0};
    unsigned int total = 0;
    for (Constituency e: list) {
        for (unsigned int i=0; i<3; ++i){
            total_votes[i] += e._party_scores[i];
        }
        total += e._num_voter;
    }

    unsigned int prop_results[3] = {0,0,0};
    for (unsigned int i=0; i<3; ++i){
        prop_results[i] += (total_votes[i]*6)/total;
        // '/' operator results 'quotient', so if we apply it first, the calculation become 0
    }

    unsigned int remained_seat = 6 -(prop_results[0]+prop_results[1]+prop_results[2]); // If there is a remained seat, we will give each of it in order of the received votes.
    while (remained_seat != 0){
        unsigned int x = std::max(total_votes[0], std::max(total_votes[1], total_votes[2]));
        if (x == total_votes[0]) {prop_results[0]+=1; total_votes[0]=0;}
        else if (x == total_votes[1]) {prop_results[1]+=1; total_votes[1]=0;}
        else if (x == total_votes[2]) {prop_results[2]+=1; total_votes[2]=0;}
        remained_seat -= 1;
    }

    run_test(plur_results, prop_results);

    //Finally, we write the post-processing data to the output file
    Out_File_Stream outfile;
    outfile.write_results(outfilePath, &list, plur_results, prop_results);
}
```

Program code: test.hpp



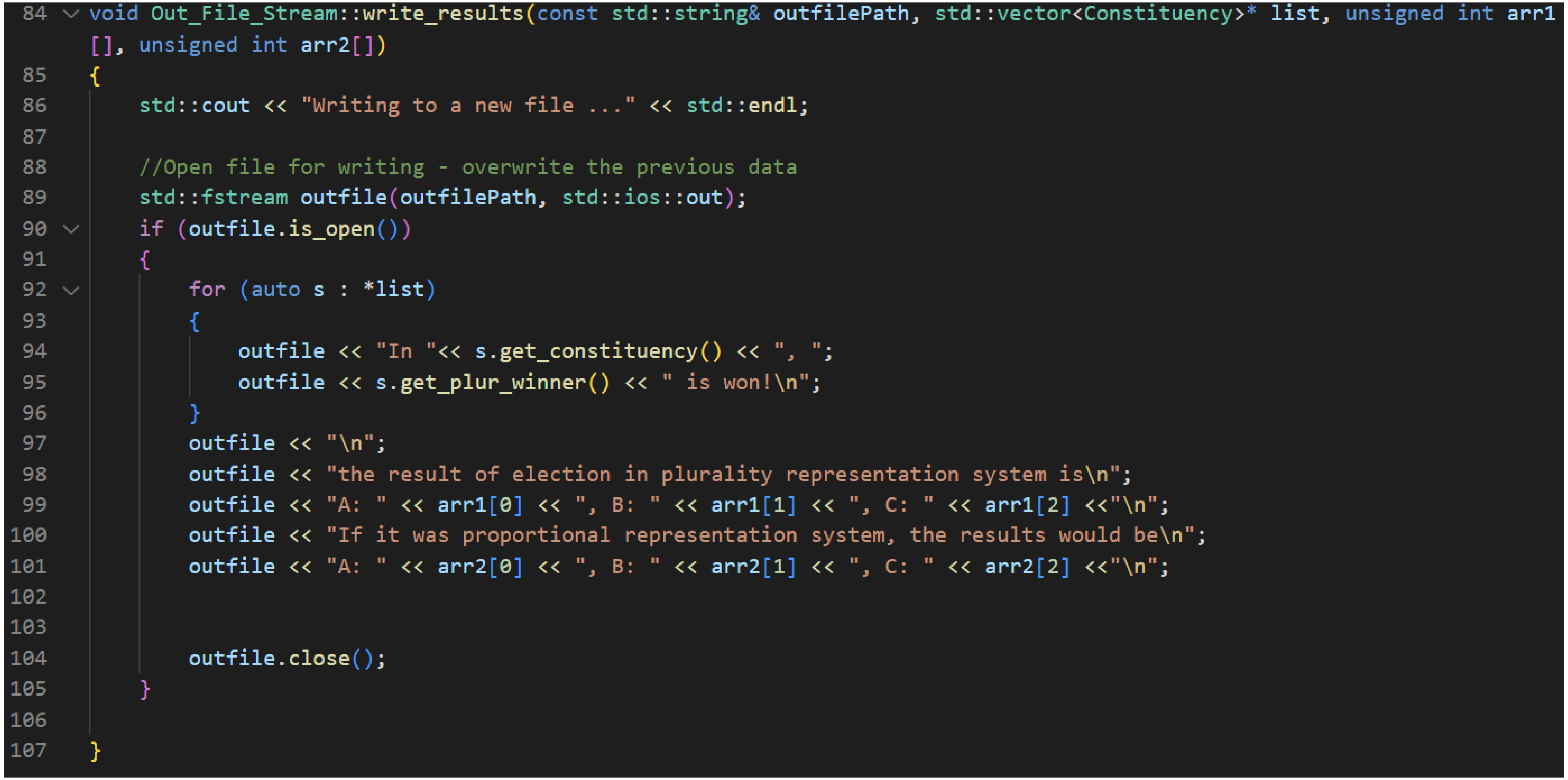
```
bool run_test(unsigned int arr1[], unsigned int arr2[]){

    bool all_passed = true;

    // initialize input array
    unsigned int correct_pulr[3] = {4, 2, 0};
    unsigned int correct_prop[3] = {2, 3, 1};

    // test each election system
    for (unsigned int i=0; i<3; ++i){
        if (arr1[i] != correct_pulr[i]) {
            all_passed = false;
            std::cout<<"test is failed for plurality system\n";
        }
        if (arr2[i] != correct_prop[i]) {
            all_passed = false;
            std::cout<<"test is failed for proportional system\n";
        }
    }

    if (all_passed==true) {std::cout<<"all test is passed\n";}

    return all_passed;
}
```

Program code: File_Stream_Prog.hpp & .cpp



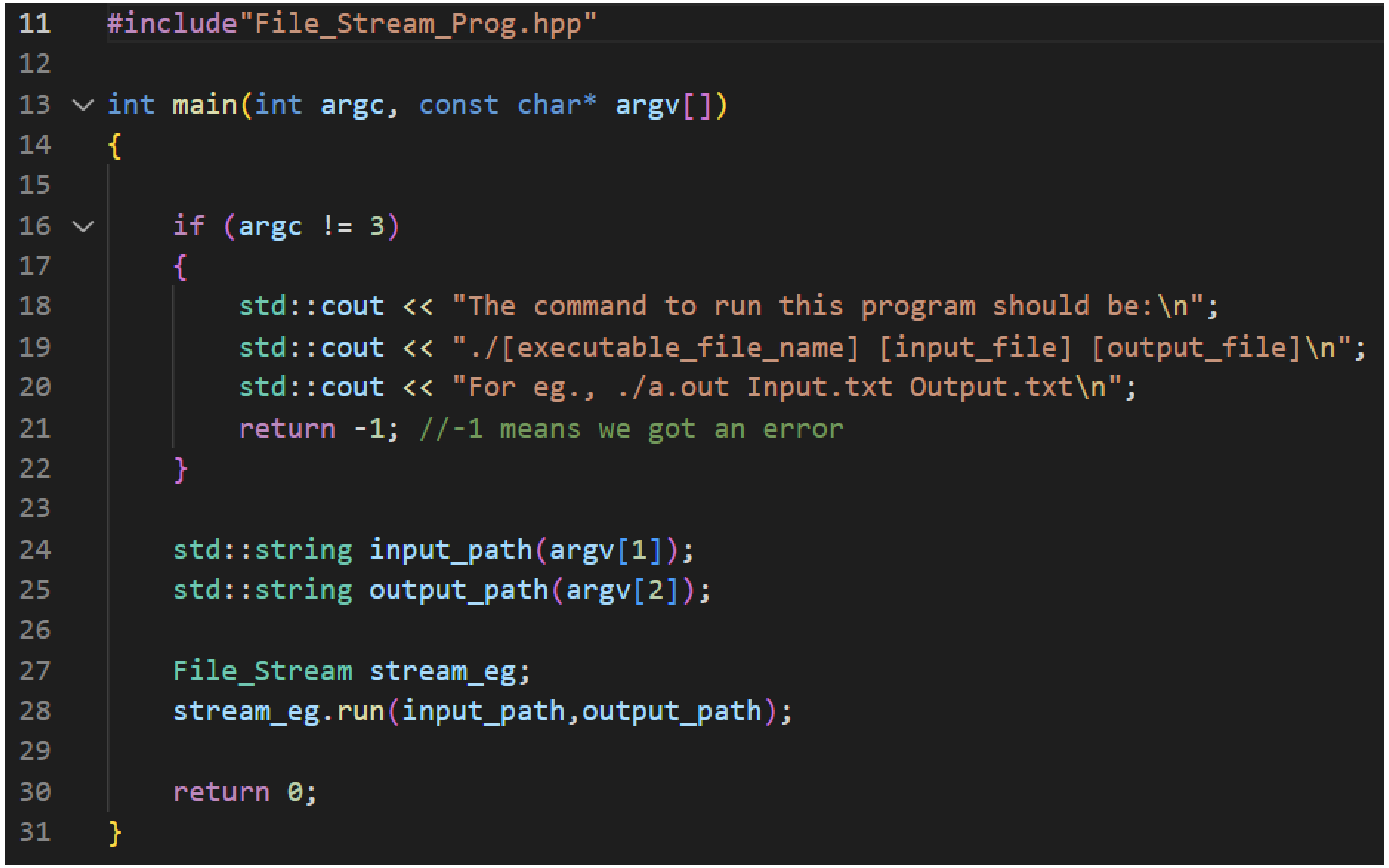
```cpp
void In_File_Stream::read_info(const std::string& infilePath,
                               std::vector<Constituency> &list)
{
    std::cout << "Reading from the input file ..." << std::endl;

    //Open file for reading
    std::fstream infile(infilePath, std::ios::in);
    if (infile.is_open())
    {
        std::string line;
        while (std::getline(infile,line))
        {
            //Load the information in each line
            //the user-defined constructor will read and parse the information.
            Constituency s(line);

            //We then store this piece of information to a list for later use
            list.push_back(s);
        }
        infile.close();
    }

    std::cout << "Finish Reading ..." << std::endl;
}
```

Program code: File_Stream_Prog.hpp & .cpp



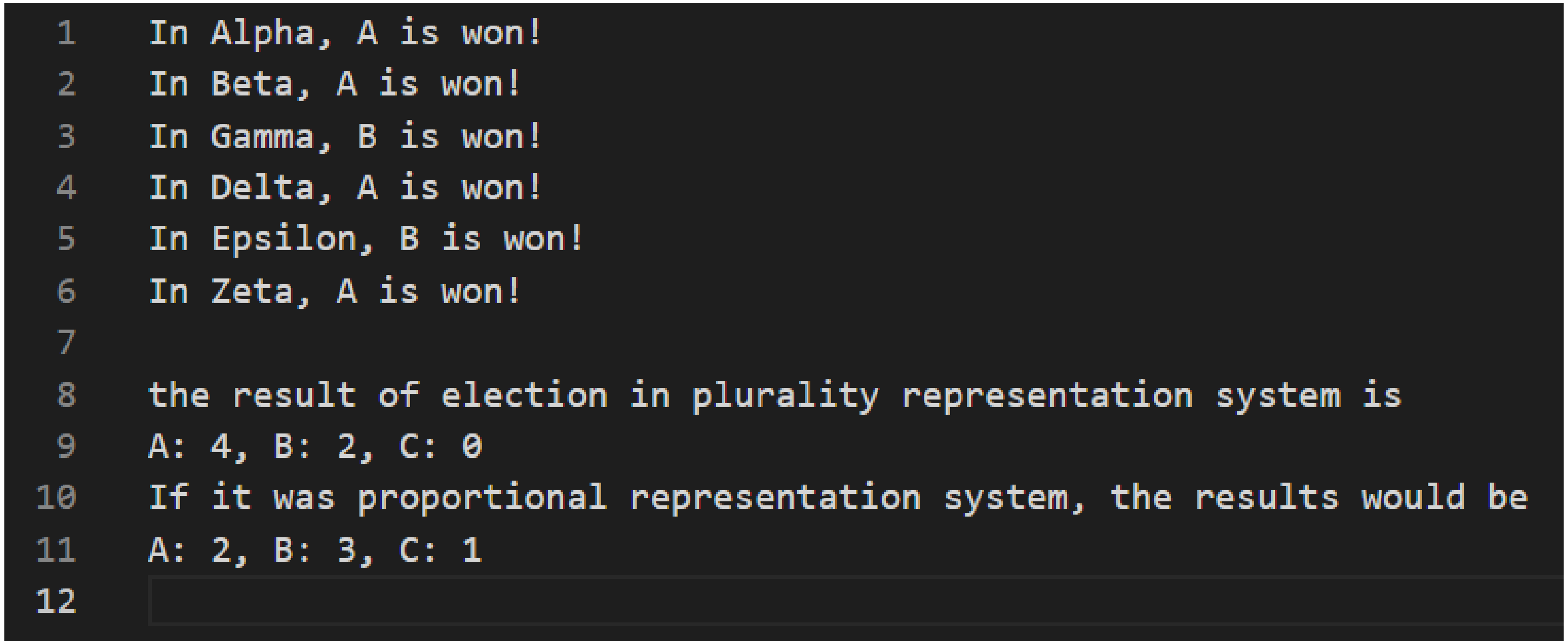
```cpp
void Out_File_Stream::write_results(const std::string& outfilePath, std::vector<Constituency>* list, unsigned int arr1
[], unsigned int arr2[])
{
    std::cout << "Writing to a new file ..." << std::endl;

    //Open file for writing - overwrite the previous data
    std::fstream outfile(outfilePath, std::ios::out);
    if (outfile.is_open())
    {
        for (auto s : *list)
        {
            outfile << "In "<< s.get_constituency() << ", ";
            outfile << s.get_plur_winner() << " is won!\n";
        }
        outfile << "\n";
        outfile << "the result of election in plurality representation system is\n";
        outfile << "A: " << arr1[0] << ", B: " << arr1[1] << ", C: " << arr1[2] <<"\n";
        outfile << "If it was proportional representation system, the results would be\n";
        outfile << "A: " << arr2[0] << ", B: " << arr2[1] << ", C: " << arr2[2] <<"\n";


        outfile.close();
    }

}
```

Program code: main.cpp



```cpp
#include"File_Stream_Prog.hpp"

int main(int argc, const char* argv[])
{

    if (argc != 3)
    {
        std::cout << "The command to run this program should be:\n";
        std::cout << "./[executable_file_name] [input_file] [output_file]\n";
        std::cout << "For eg., ./a.out Input.txt Output.txt\n";
        return -1; //-1 means we got an error
    }

    std::string input_path(argv[1]);
    std::string output_path(argv[2]);

    File_Stream stream_eg;
    stream_eg.run(input_path,output_path);

    return 0;
}
```

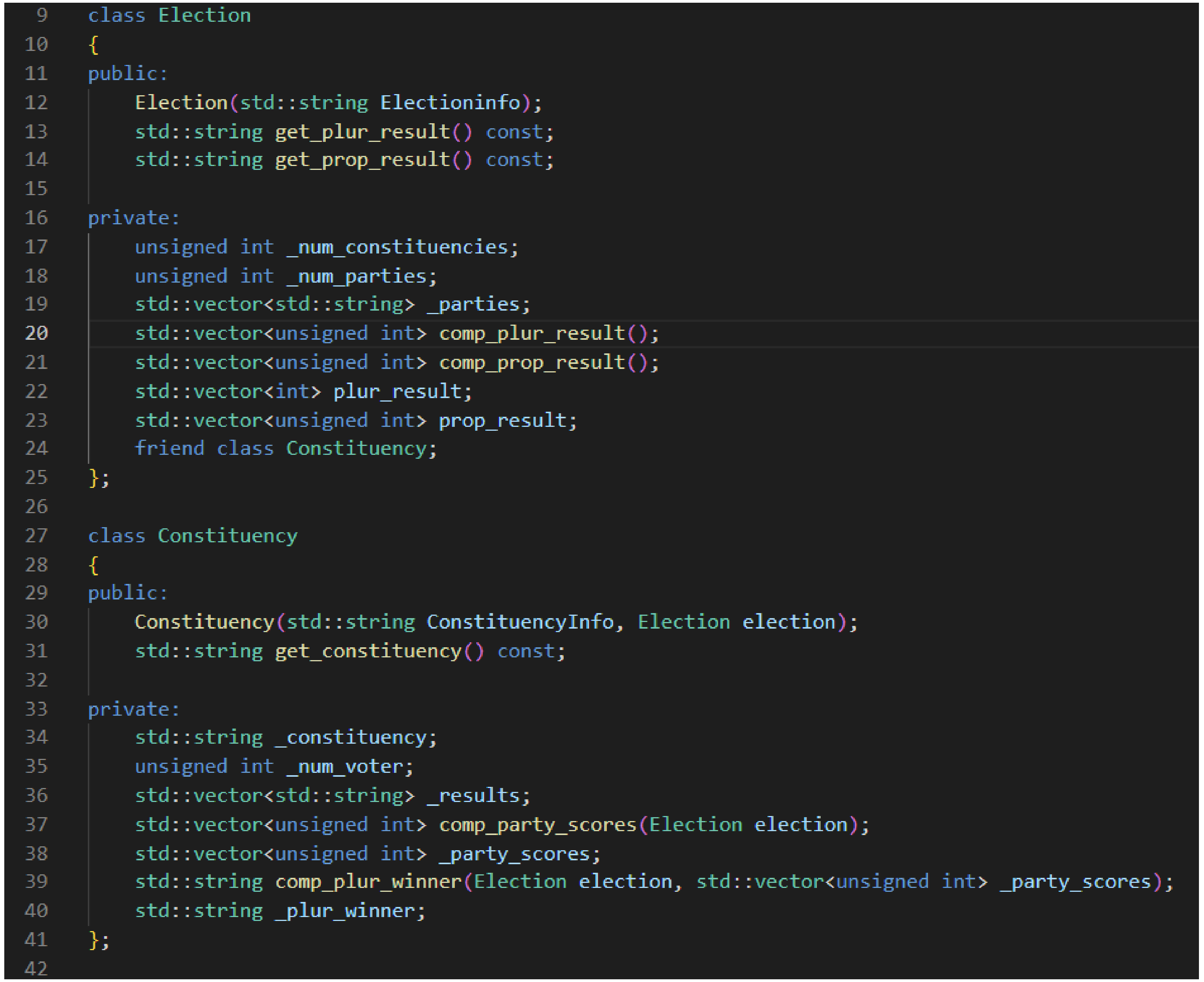
Output file



```
In Alpha, A is won!
In Beta, A is won!
In Gamma, B is won!
In Delta, A is won!
In Epsilon, B is won!
In Zeta, A is won!

the result of election in plurality representation system is
A: 4, B: 2, C: 0
If it was proportional representation system, the results would be
A: 2, B: 3, C: 1

```

My failures

At first, I tried to create two classes and put election information into the Election class.
However, the calculation between the two classes was so complicated that I quit.



```cpp
class Election
{
public:
    Election(std::string Electioninfo);
    std::string get_plur_result() const;
    std::string get_prop_result() const;

private:
    unsigned int _num_constituencies;
    unsigned int _num_parties;
    std::vector<std::string> _parties;
    std::vector<unsigned int> comp_plur_result();
    std::vector<unsigned int> comp_prop_result();
    std::vector<int> plur_result;
    std::vector<unsigned int> prop_result;
    friend class Constituency;
};

class Constituency
{
public:
    Constituency(std::string ConstituencyInfo, Election election);
    std::string get_constituency() const;

private:
    std::string _constituency;
    unsigned int _num_voter;
    std::vector<std::string> _results;
    std::vector<unsigned int> comp_party_scores(Election election);
    std::vector<unsigned int> _party_scores;
    std::string comp_plur_winner(Election election, std::vector<unsigned int> _party_scores);
    std::string _plur_winner;
};

```

계산과학 이론 및 실습2

THANK YOU

made by Jeongwoo Kim