# Vote system

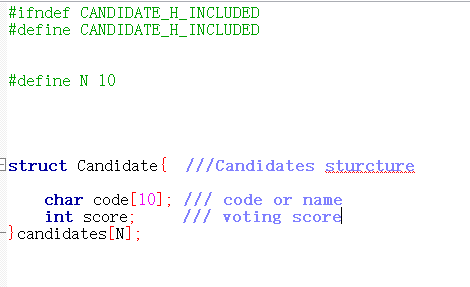
## System design

The system is designed with two types of inputs, integer and strings. For the integer it’s the number of candidates and the number of voters, the string type of input corresponds to the name or code ID of the candidates. After the integer inputs the system computes by incrementing the score of the corresponding name inputted during the vote, if the input doesn’t correspond to any candidate the system outputs a warning letting the voter know that the vote has not been taken into account due to the invalid code or candidate name. The bubble sorting algorithm is afterwards used to sort the score of the candidates from the highest to the lowest and the results are outputted. Furthermore if in any case there is two winners then the system will let the user know that the candidates should vote again.

### Detailed design

#### Candidate design

For the candidate units, **structures** has been used with two inside variables: the code which correspond to the name or ID of the candidate and the score which represent the points gained during the vote. An array of candidates is first created with a maximum settled at 10 candidates when the program runs. The implementation is done in a header file, Figure 1 gives an overview of the candidate.h file



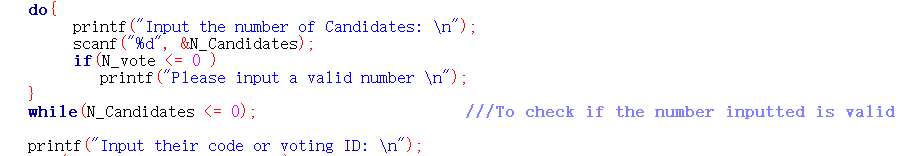
**Fig.1: Candidate.h file**

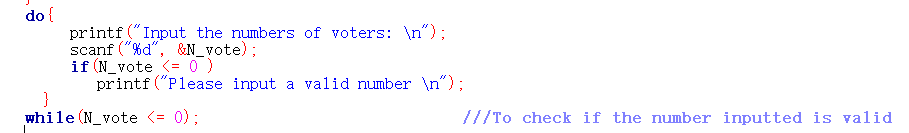
#### The inputs:

As mentioned in the previous sections, there are two type of inputs in the system. The first one being the integer type of input correspond to the number of candidates, and the number of voters.

Both inputs are needed to be positive numbers for the system to run correctly, otherwise it will loop on that step until a valid number has been inputted.

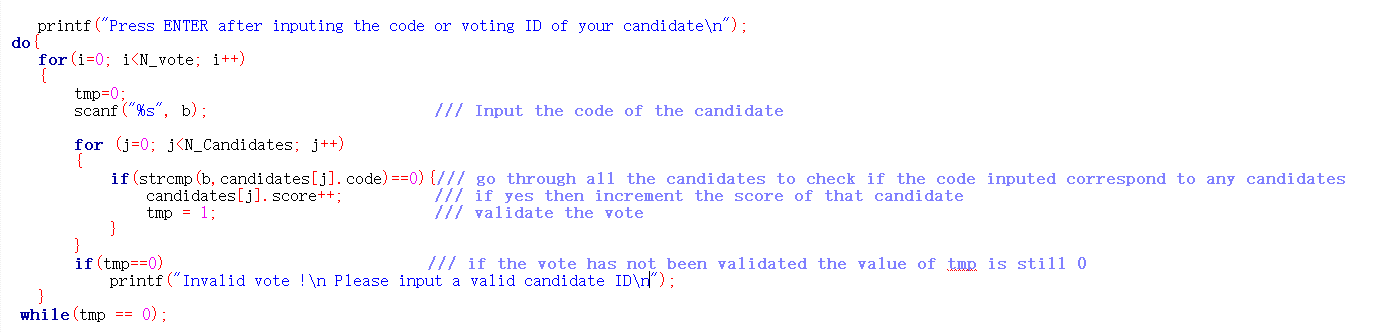
However both integer type of inputs are done successively. After entering the number of candidates, the user has to input the second type which is a string type corresponding to the code or ID name of the candidate. That code is later used by the voters to choose the person they want to vote for. During the vote, the ID inputted is compared to the ID of the candidate previously entered using the function **strcmp.** If there is any match then the score of the corresponding candidate is incremented.





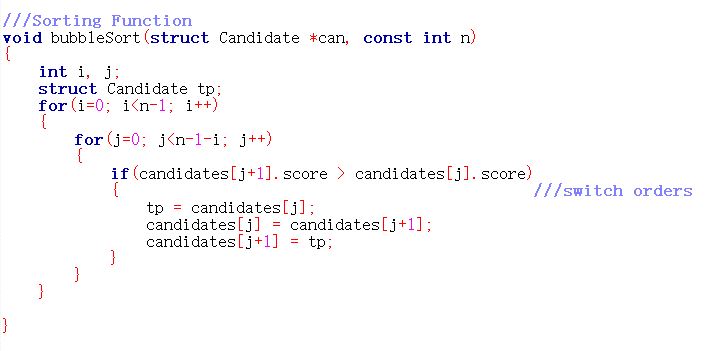
**Fig. 2: overview of input Integers**

If during the vote, a code doesn’t correspond to any candidate the system will warn the voter showing that the vote is invalid and will loop until the voter inputs a valid ID.



**Fig. 3: Implementation of the voting process**

After the vote, the bubble sorting algorithm is used to sort the candidate from the one who has the highest points to the one with the lowest then print the results. Furthermore if there is two winners, in others words two top candidates with the same score, then the system will ask them to proceed to a second vote.



**Fig. 4 Bubble sort Algorithm implementation**