SAE 1.01 : Système de vote

Sommaire :

Vote majoritaire:

Un vote majoritaire, ou scrutin majoritaire est un système électoral qui octroie la victoire au candidat ayant le plus de votes. Ce système de vote a des possibilités d'égalité lorsque 2 jeux se retrouvent avec le même nombre de voix.

Pour réaliser ce vote en C++, on a d'abord commencé par initialiser les variables, qui sont : les variables d'entier qui contiennent le nombre de voix du jeu-vidéo et les votes blancs, une variable qui sert à compter le numéro des lignes ainsi que la variable qui sert à contenir une ligne du fichier.

Tant que la fin du fichier n'est pas atteinte, la ligne lue est stockée dans une chaîne de caractère. Si le numéro de la ligne est un multiple de 3, alors on incrémente les variables de 1 en fonction des votes.

En effet, si le vote est le caractère 1, alors on incrémente la variable VoteCSGO de 1.

Si le vote est le caractère 2, alors on incrémente la variable VoteSFII de 1.

Si le vote est le caractère 3, alors on incrémente la variable VoteCIV6 de 1.

Si le vote est le caractère 4, alors on incrémente la variable VoteMario de 1.

Pour finir, si le vote n’est pas un caractère compris entre 1 et 4, alors on incrémente la variable VoteB de 1.

Une fois les voix comptées et respectivement stockées dans les variables des jeux ou du vote blanc, nous faisons appel à une fonction annexe, nommée tableau, qui prend en paramètre les variables contenant les voix et renvoie un vecteur de couple, composé d’une chaine de caractère, qui est le nom du jeu, et de la variable d’entier, qui est le nombre de voix du jeu respectif. Ce vecteur est trié dans l’ordre croissant des votes.

Une fois ce vecteur en notre possession, nous pouvons écrire dans le fichier les résultats à l’aide de cout.

Pondération ternaire/Vote Négatif :

La pondération ternaire est un système de vote similaire au système de vote majoritaire, mais il présente une différence importante. En effet, si le votant ne trouve pas de proposition qui lui plaise, il peut voter contre la proposition dont il veut le moins. Ce vote, que l’on peut appeler vote noir (par analogie au vote blanc) ou encore vote négatif, est comptabilisé comme un −1 dans le total des votes, là où un vote normal compterait comme un +1.

De part sa ressemblance avec le vote majoritaire, le programme en C++ ressemble également au programme du vote majoritaire :
Les mêmes variables sont initialisées et utilisées, la même méthode pour trouver les votes dans le fichier, etc. La différence se trouve dans la boucle qui sert à ajouter les votes dans les variables : Avant de rajouter une voix à une variable, il vérifie s’il y a un moins (‘-’) devant, auquel cas il enlève le vote à la variable.
Ce programme utilise aussi une fonction similaire à la fonction tableau() du système majoritaire, puis écrit dans le fichier de sortie les résultats obtenus.

Élection à 2 tours (élection présidentiel française) :

Le système de vote à 2 tours comme celui dans les présidentielles françaises repose sur le principe de 2 tours, le premier tour fait ressortir 2 personnes qui ont eu le plus de vote qui s’affronteront au second tour ou il y aura un gagnant.

Ce système de vote a quelque spécificité, il y a une possibilité de voter blanc.
Il est aussi possible de pouvoir gagner avec la majorité absolue au premier tour.

Il y a des possibilités d'égalité, par exemple au premier tour si 3 personnes terminent premières à égalité alors il faut refaire le vote, de même pour le second tour si les 2 candidats arrivent à une égalité parfaite alors il faudra procéder à un nouveau vote.
Il y a aussi des possibilités d'événement telle que le vote blanc majoritaire au premier tour : si cela arrive une nouvelle élection aura lieu mais aucun des candidats s'étant présenté aux élections ne pourront se représenter. Il y a aussi une possibilité que le vote blanc arrivé majoritaire au second tour dans quel cas il faudra refaire les élections.

Pour ce code j'ai utilisé des structs, une pour les votants, pour stocker leur information : nom, prenom, votetour 1, votetour 2.
Et aussi une struct pour les jeux afin de stocker leur nom et le nombre de votes qu'ils ont reçu, il y en aura deux une pour le premier tour et une pour le second tour.

Dans le code j'ai commencé par initialiser maj qui me servira pour détecter si il y a une majorité absolue au premier tour, puis je crée la struct ListeJeux1tour qui me servira pour le premier tour et l'initialiser grâce à la fonction initaliseJeux1tour qui permettra de mettre les noms des jeux dans la struct et de mettre leur nombre de vote a 0.

Puis je crée la struct ListeVotant qui stockera les informations concernant chaque votant, j'utilise donc la fonction ajoutInfoDansVecteur qui parcourra le fichier d'entrée et ajoutera les informations des votants.

Pour compter les votes nous allons utiliser comptabilisationvotes qui parcourra les vote du premier tour et ajoutera au nombre de vote de chaque candidat si on vote pour eux, ensuite nous utiliserons la fonction sort qui triera les résultats afin de savoir qui a eu le plus de vote.

Puis nous utiliserons la fonction Sortie1tour qui donnera les résultats du premier tour et comptabilise les exceptions citées auparavant et renverra 1 dans la variable maj en cas de majorité absolue, dans quel cas le second tour n'a pas lieu.

Si la variable maj n'est pas égale à 1 donc s'il n'y a pas de majorité au premier tour alors le second tour commence.

Pour le second tour, nous commençons à créer la struct ListeJeux2Tour, puis nous initialiserons avec la fonction initListeJeux2tour qui récupère le nom des 2 gagnants et leur nombre de vote égale à 0, nous utiliserons la fonction comptabilisation Votes qui regarde si le nombre de candidats est égale à 2, si c'est le cas alors il récupérera les votes des votants pour le second tour et augmentera le nombre de vote de chaque candidat si on vote pour eux.

Grâce à la fonction sort les résultats triés sont envoyés dans la fonction Sortie2tour qui affiche les résultats du second tour en prenant en compte les éventualités citées au début.

Vote “classement” :

Fonctionnement du système de vote :
Chaque votant fait un classement des 4 jeux, en commençant par son préféré jusqu’à celui qu’il aime le moins. Le premier jeu (donc son jeu préféré parmi la liste) reçoit 3 points en plus, le deuxième jeu reçoit 2 points, le troisième 1 point et le dernier (donc le jeu qu’il aime le moins dans la liste) ne reçoit pas de points. Tous les points sont comptabilisés et le jeu avec le plus de points est donc le jeu préféré du groupe de votants (ici, la promo).

Fonctionnement du programme :
Tout d’abord, nous incluons différentes bibliothèques nécessaires au bon fonctionnement du programme : iostream, vector (pour ajouter les tableaux), iomanip et algorithm (pour la commande sort).
On déclare ensuite deux structs : Jeux et InfoVotant, le premier est une struct contenant le nom et le nombre de points d’un jeu sous la forme d’un string et d’un entier naturel, le second contient le nom, le prénom et les votes du votant sous la forme de deux strings et d’un tableau d’entiers naturels.

initListeJeux :
Pour utiliser cette fonction, nous devons d’abord initialiser un tableau de structs <Jeux>, ce qui donne un tableau à deux dimensions où chaque case est dédiée à un jeu. Cette fonction modifie ce tableau de telle sorte à ce qu’il contienne les 4 jeux sujets au vote : Counter-Strike : Global Offensive, Street Fighter II, Civilization VI et Mario Kart.
La fonction réalise cela en redimensionnant le tableau de 4 cases et en ajoutant le nom du jeu dans chaque case (de 0 à 3), puis de mettre le nombre de points de chaque jeu à 0.

ajoutInfosDansVecteur:
Pour utiliser cette fonction, nous devons d’abord initialiser un tableau de structs <InfoVotant>, ce qui donne un tableau à deux dimensions où chaque case est dédiée à un votant/élève. Cette fonction modifie ce tableau en injectant les informations de tous les votants à partir du fichier entrée par le biais de la fonction “getline()”. Chaque ligne est extraite du fichier une par une avec un pointeur chargé de garder en mémoire le nombre de lignes parcourus :

- La première ligne est injectée (sous forme de string) dans le Nom d’une nouvelle case du tableau.

- La seconde est injectée (sous forme de string) dans le Prénom de la case du tableau précédemment créé.
- La troisième (qui se présente sous la forme d'un nombre composé de 4 chiffres de 1 à 4) est prise caractère par caractère et chaque caractère est converti en son équivalent en entier naturel (char '1' devient unsigned 1, '2' devient unsigned 2, etc). Le pointeur est ensuite réinitialisé à 0.

Et cela continue jusqu'à que la condition "cin.eof()" soit vraie donc que l'on ait atteint la fin du fichier.
On se retrouve donc à la fin avec un tableau de <InfoVotant> d'une taille égale au nombre de votants participants au vote.

comptabilisationVotes :
Cette fonction permet de comptabiliser les votes contenus dans le ListeVotant et de les ajouter dans le tableau ListeJeux.
Pour cela la fonction parcourt l'intégralité du tableau ListeVotant et en particulier la case VOTES où est stocké le classement d'un élève. Le jeu contenu (caractérisé par un nombre allant de 1 à 4) dans la première case du tableau VOTES ajoute 3 à nbPoints du jeu correspondant dans vecteur <Jeux> ListeJeux, la deuxième case ajoute 2, la troisième case ajoute 1 et la dernière n'ajoute rien. Et cela pour tous les participants au vote. A la fin de cette fonction, tous les votes sont comptabilisés.

sort(ListeJeux.begin(), ListeJeux.end(), orderBynbPoints):
Cette fonction permet de trier le tableau ListeJeux par le nombre de points dans un ordre décroissant, cela est permis par la fonction intermédiaire orderBynbPoints qui renvoie un booléen et que nous avons défini. A la fin de cette fonction, nous avons tous les Jeux du tableau qui sont triés du plus populaire au moins populaire.

Sortie :
Cette fonction permet d'envoyer sur la sortie standard cout les résultats des votes. Elle donne un classement des jeux du plus populaire au moins populaire en ajoutant aussi le nombre de points de chaque jeu. Cette fonction prend aussi les cas où :

- Les deux jeux les plus populaires ont le même nombre de points, ils sont alors à égalité et gagnent tous deux le vote.
- Les trois jeux les plus populaires ont le même nombre de points, ils sont alors tous les trois à égalité et gagnent le vote.
- Les quatre jeux ont le même nombre de points, la promo n'a donc pas pu se décider et elle aime les 4 jeux de manière égale.