Correction SN Programmation

Exercice 1 :

1. Donnons la structure de base d'un programme C++ :

```
#include<iostream>
using namespace std;
int main ()
{
        …
  return 0;
}
```

2.

3. Donnons le rôle de using namespace std :
   Elle permet de ne pas répète std et gagner en temps car en son absence il faut systématiquement utiliser le préfixe std :: devant tous les éléments qui en sont issus *(exemple std ::cout<<''entrez une valeur''<<).*

4. OUI le langage C++ est un POO (Programmation Orienté Object).

5. Un IDE (Editeur de code source) : Est un éditeur de texte qui aide à la rédaction du code logiciel à travers la saisie automatique en fonction du langage et la vérification de bug dans le code pendant la rédaction.
   Exemple : Visual Studio Code, Sublime Text, Notepad ++.

Exercice 2 :

A- Faire un programme en C++ qui demande a l'utilisateur **d'entrer 10 notes** puis **l'affiche** et **calcul sa moyenne.** Après elle donne la **plus grande note et sa position** avant de les afficher par **ordre croissant**.

NB: Vous utilisez une fonction qui va faire le classement et une autre qui va trouver la position de la plus grande note

Solution

Etape 1: Compréhension du problème

Pour résoudre ce problème il faut tout d'abord comprendre les différentes taches qui entre en jeux dans la programme:

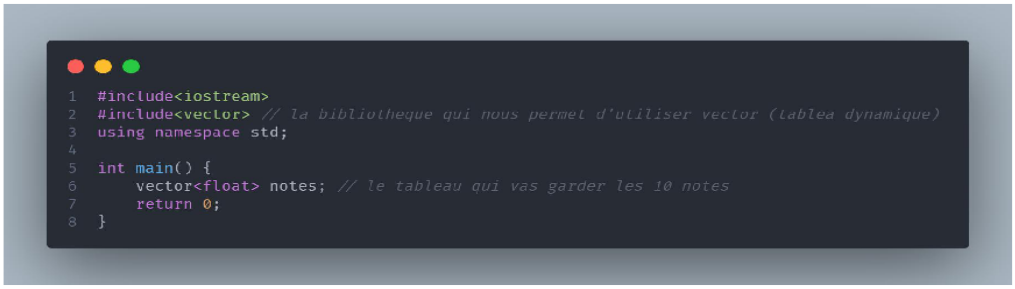
- Récupérer les 10 notes de utilisateurs
- Afficher ces 10 notes
- Calculer la moyenne de l'utilisateur en fonctions de ces 10 notes.
- Trouver la position de la plus grande note en fonction de la quel on poura trouver cet dernier
- Enfin classer les notes par ordre croissant (du plus petit au plus grand).

Et donc ce que nous allons faire c'est que pour chaques taches ci dessu nous allons faire une fonction qui realise la tache, et a la fin nous aurons tous les outils pour resoudre le probleme.

Etape 2: Traitement des différentes sous taches du problème

a) Une fonction qui récupère les 10 notes de l'utilisateur:

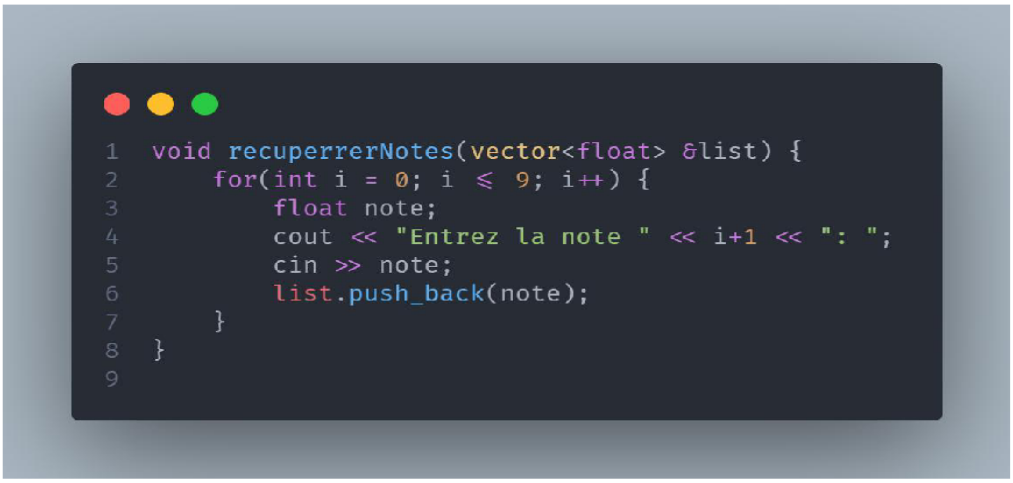
On commence par créer une variable pour garder les 10 notes (un tableau de préférence)



```cpp
#include<iostream>
#include<vector> // la bibliotheque qui nous permet d'utiliser vector (tablea dynamique)
using namespace std;

int main() {
    vector<float> notes; // le tableau qui vas garder les 10 notes
    return 0;
}
```

En suite on crai une fonction qui vas prendre ce tableau comme argument pour ensuite y inserer les notes de l'utilisateure

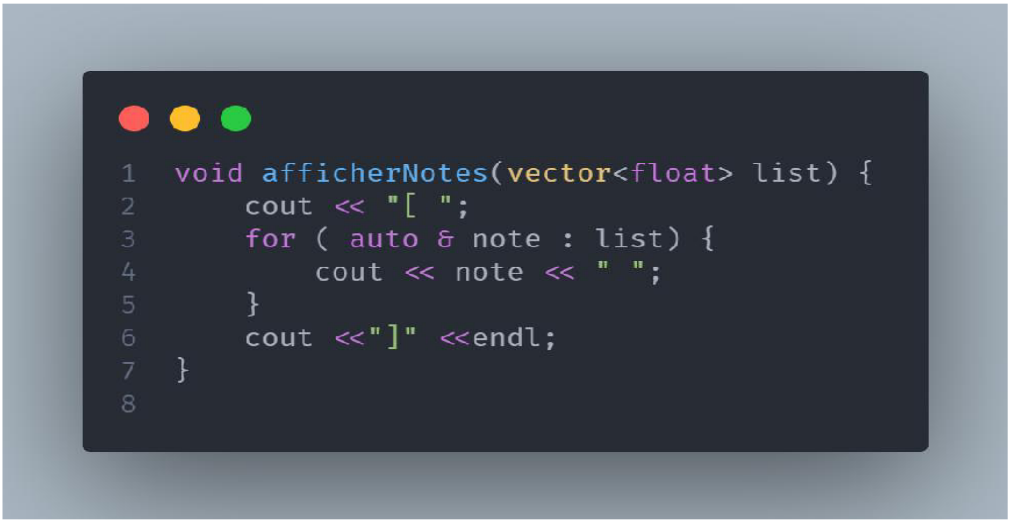


```cpp
void recuperrerNotes(vector<float> &list) {
    for(int i = 0; i <= 9; i++) {
        float note;
        cout << "Entrez la note " << i+1 << ": ";
        cin >> note;
        list.push_back(note);
    }
}
```

Dans le code ci dessus, crée une fonction `recuperrerNotes` et on lui donne comme paramétré un vecteur passé par référence (avec `&`) appelé `list`
Puis on cree une boucle pour qui vas iterer de 0 a 9 (10 fois) pour chaques noms, et a chaques iterations, on demande la note numero `(i+1)` a l'utilisateur puis on la stoque dans la variable `note` avant de l'ajouter dans `list`

b) Une fonction qui affiche les notes a l'ecrans

On vas adopter la meme logique que pour la fonction precedente, saufe que cet fois ci pour chaque iteration on affiche juste l'element de la liste



```cpp
void afficherNotes(vector<float> list) {
    cout << "[ ";
    for ( auto & note : list) {
        cout << note << " ";
    }
    cout <<"]" <<endl;
}
```

Dans le code si dessus on voit une variation de la boucle pour (for). Ce type de boucle est utilisé pour itérer a travers les variables de type vector. Ici on note 3 clés (auto, "&", ":") ces clé une fois combiner ont la sémantique qui est

"Traverse automatiquement le tableau qui est après les colonnes (:) et a chaque traversée (itération) transfert le contenu de chaque élément du tableau a la variable tampons qui est âpres esperluette (&) "

Donc notre condition dans la boucle dit littéralement "A chaque fois que tu traverse un élément du tableau `list`, transferts son contenue dans la variable `note`"

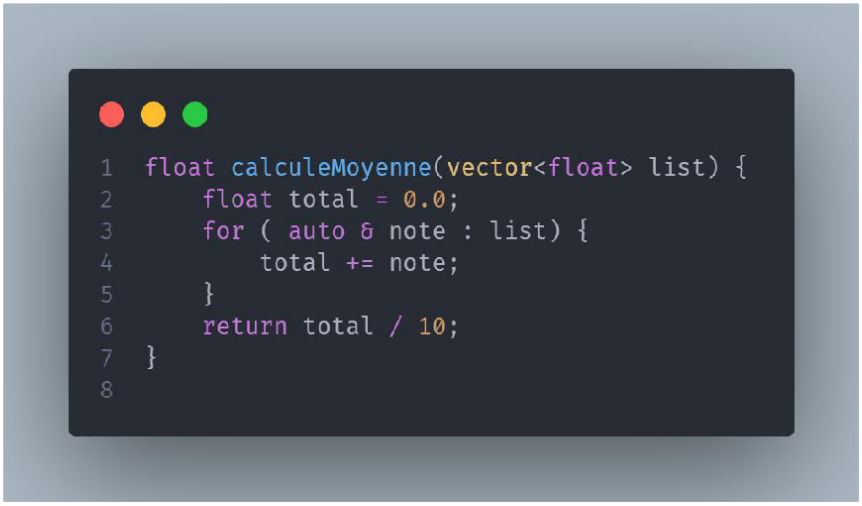
Puis après on as qu'a afficher le contenue de note.

NB: "[" et "]" sont afficher avant et après le contenue du tableau pars souci de présentation donc sont inutile et peuvent être omissent.

Egalement dans cet fonction je ne passe pas `list` par reference par ce que je n'ait pas besoin de modifier son contenue originale.

J) Calculer la moyenne en fonction des notes

Pour ce faire nous devons tout d'abord calculer la somme de tous les 10 notes et ensuite diviser cet somme la par 10 pour obtenire la moyenne.
Nous allons donc creer une fonction qui prend un vector comme parametre et retourne un float.



```
float calculeMoyenne(vector<float> list) {
    float total = 0.0;
    for ( auto & note : list) {
        total += note;
    }
    return total / 10;
}

```

On commence par initialiser total par 0.0 vue que le 0 en addition est neutre. Puis pour chaque element du tableau list, on vas ajouter la valeur de cet element (note) au total. Et enfin, on retourne juste la moyenne qui est le total / 10

d) Trouver la position de la plus grande note.

Pour effectuer cet tache on peut procéder de manier sequencielle, c'est a dire on vas ce fixer une position que l'on considéré comme l'indice de la plus grande note (de préférence celle la premier valeur du tableau). Et en suite on vas comparer cet valeurs de cet position la avec tous les éléments du tableau pour dénicher l'indice de la plus grande valeurs.

```cpp
int plusGrandeNote(vector<float> list) {
    int pos = 0;
    for (int i = pos + 1; i <= 9; i++) {
        if (list[i] > list[pos]) {
            pos = i;
        }
    }
    return pos;
}
```

Donc la variable `pos` ce fixe une valeur de 0. Pour chaque itération on compare l'element de l'indice `pos` avec l'element `i` du tableau. Et quand l'element `i` est plus grand que l'element `pos`, cela veut dire que l'element i est la nouvel plus grande valeurs donc `pos = i` .
Et a la fin on retourne `pos` qui cera donc l'indice de la plus grande valeurs.

e) classer les notes par ordre croissant (du plus petit au plus grand):

Pour classer des éléments par ordre croissant ou décroissants on utilise des algorithmes spécifiques appelée algorithmes de trie. Parmi ces algorithmes on a l'algorithme du trie a bulle qui est le plus simple et qui marche bien dans notre cas avec les petites liste.

- Le tri à bulle est un algorithme de tri simple pour classer les éléments d'une liste.
- Il compare les éléments voisins et échange les plus grands/plus petits jusqu'à ce qu'ils soient triés.
- Ce processus est répété plusieurs fois jusqu'à ce que la liste soit triée.

```cpp
void classerNotes(vector<float> &list) {
    for(int i = 0; i <= 9; i++) {
        for (int j = 0; j <= 8; j++) {
            float temp;
            if (list[j] > list[j+1]) {
                temp = list[j];
                list[j] = list[j+1];
                list[j+1] = temp;
            }
        }
    }
}
```

On remarque que ici le classement effectuée dans la deuxième boucle. Ou des lors qu'on ce rend compte qu'un élément est plus grand que le suivant (la condition du `if`)

Etape 3: Assembler les différents fonctions pour résoudre le problème

Pour cet dernier étape, puisque nous avons tous les outils nécessaire (fonctions), nous pouvons donc appeler ces fonctions la dans la fonction main, pour pouvoir résoudre le problème. On note que a chaque fois que l'on modifie le tableau on l'affiche ensuite pour montrer son nouveau contenue.

```cpp
int main() {
    vector<float> notes;

    recuperrerNotes(notes);
    cout << "voici vos notes: " << endl;
    afficherNotes(notes);

    cout<< "La moyenne est: " << calculeMoyenne(notes);

    float plusGrande = plusGrandeNote(notes);
    cout << "la plus grande note est la note numero: ";
    cout<< plusGrande<< " et c'est : "<< notes[plusGrande] << endl;

    classerNotes(notes);
    cout << "voici vos notes une fois classer: " << endl;
    afficherNotes(notes);
    return 0;
}

```

Donc apres avoir fait tout ca le programme finale ressemble a

```cpp
#include<iostream>
#include<vector>
using namespace std;

void recuperrerNotes(vector<float> &list) {
    for(int i = 0; i <= 9; i++) {
        float note;
        cout << "Entrez la note " << i+1 << ": ";
        cin >> note;
        list.push_back(note);
    }
}

void afficherNotes(vector<float> list) {
    cout << "[ ";
    for ( auto & note : list) {
        cout << note << " ";
    }
    cout <<"]" <<endl;
}

float calculeMoyenne(vector<float> list) {
    float total = 0.0;
    for ( auto & note : list) {
        total += note;
    }
    return total / 10;
}

int plusGrandeNote(vector<float> list) {
    int pos = 0;
    for (int i = pos + 1; i <= 9; i++) {
        if (list[i] > list[pos]) {
            pos = i;
        }
    }
    return pos;
}

void classerNotes(vector<float> &list) {
    for(int i = 0; i <= 9; i++) {
        for (int j = 0; j <= 8; j++) {
            float temp;
            if (list[j] > list[j+1]) {
                temp = list[j];
                list[j] = list[j+1];
                list[j+1] = temp;
            }
        }
    }
}

int main() {
    vector<float> notes;

    recuperrerNotes(notes);
    cout << "voici vos notes: " << endl;
    afficherNotes(notes);

    cout<< "La moyenne est: " << calculeMoyenne(notes);

    float plusGrande = plusGrandeNote(notes);
    cout << "la plus grande note est la note numero: ";
    cout<< plusGrande<< " et c'est : "<< notes[plusGrande] << endl;

    classerNotes(notes);
    cout << "voici vos notes une fois classer: " << endl;
    afficherNotes(notes);
    return 0;
}
```

Exercice 3 :

1. Il s'agit du langage C.
2. Le programme demande à l'utilisateur un ensemble d'élément puis calcul et affiche le nombre d'éléments positifs, négatifs, paires, impaires puis le max et le minimum.
3. Il y a 03 variables d'entrées : tab[400], N, i.
4. Il y a 06 variables de sorties : a, b, c, d, min, max.
5. Nommons les lignes qui contiennent des erreurs :
   3 ; 5 ; 11 ; 16 ; 23 ; 24 ; 28 ; 33 ; 45 ; 46 ;