

Nom	Prénom	Matricule	Programme
Dorval	Arnaud	111 155 275	Génie Informatique

Travail Pratique 1

Rapport d'investigation:

Cahier des charges

Le programme principal prend des valeurs numériques en saisi jusqu'à ce que le tableau soit rempli. Par la suite, il prend une valeur numérique en saisi et l'utilise comme « seuil ». Il calcule par rapport au seuil le pourcentage de valeurs pris en saisi au début du qui sont plus grand que celui-ci.

Stratégie

- Je vais lire chacune des lignes de code afin voir s'il y a des erreurs de syntaxes dans chacun des 3 fichiers. Il est important de corriger les erreurs de syntaxe en premier afin que le code puisse compiler un minimum afin de mieux detecter les erreurs de logique.
 - programmePrincipal.cpp
 - fonctionsUtilitaires.h
 - fonctionUtilitaires.cpp
- Je vais insérer des break point dans programmePrincipal.cpp à des endroits stratégiques afin segmenter la construction du code et faciliter la résolution des erreurs. Exemple de points stratégique pour les break point :
 - Juste avant (int main())
 - Après la déclaration des variables
 - Avant la fermeture d'une boucle for
 - À l'appel des différentes fonctions de fonctionsUtilitaires

Erreur 1

Localisation

```
programmePrincipal.cpp
12
13 #define MAX CAS =
fonctionsUtilitaires.h
11 #define MAX_CAS = 4
12
```

La première erreur est la façon utilisé pour déclarer MAX_CAS. Celle-ci a été localisé en survolant le code à la recherche d'erreure de syntaxe. (C'était une mauvaise pratique)

Solution

```
13 const int MAX CAS = 4;
```

Erreur 2

Localisation

programmePrincipal.cpp

```
13 const int MAX_CAS =  
14 const int MAX_VALEURS
```

fonctionsUtilitaires.h

```
11 #define MAX_CAS = 4  
12
```

En corrigeant la façon dont cette variable est déclaré à l'erreur on découvre une nouvelle erreur. L'erreur donné par eclipse est une erreur de mauvaise déclaration. Puisque la syntaxe est bonne, on peut conclure que cette variable est probablement déclaré ailleurs dans le code. Elle a été ainsi localisé dans fonctionsUtilitaires.h. Puisque ces deux variables sont utilisé dans utilitaire et principal ça a plus de sens de déclarer les variables dans le « .h ». (C'était une mise en garde)

Solution

fonctionsUtilitaires.h

```
12 const int MAX_CAS = 4;  
13 const int MAX_VALEURS
```

Erreur 3

Localisation

programmePrincipal.cpp

```
15 static const int MAX_VALEURS = 4;  
16
```

L'erreur est la façon utilisé pour déclarer MAX_VALEURS. Celle-ci a été localisé en survolant le code à la recherche d'erreure de syntaxe. (C'était une mauvaise pratique)

Solution

```
15 const int MAX_VALEURS = 4;  
16
```

Erreur 4

Localisation

programmePrincipal.cpp

```

24 int register;
25 for (i = 0; i < MAX CAS; ++i)
26 {
27     for (int j = 0; j < MAX VALE
28     {
29         cin >> register;
30         tabValeurs[i, j] = regis
31     }

```

L'erreur est l'utilisation du nom register comme nom de variable. Celle-ci a été localisé en survolant le code à la recherche d'erreur de syntaxe. Register est déjà une fonction utilisé en C++ et ne peut donc pas être utilisé comme nom de variable. (C'était une mise en garde à corriger)

Solution

```

24 int registre;
25 for (i = 0; i < MAX CAS; ++i)
26 {
27     for (int j = 0; j < MAX VALE
28     {
29         cin >> registre;
30         tabValeurs[i, j] = regis
31     }

```

Erreur 5

Localisation

programmePrincipal.cpp

```

25 for (i = 0; i < MAX_CAS; ++i)
26 {
27     for (int j = 0; j < MAX_VAL
28     {
29         cin >> registre;|
30         tabValeurs[i, j] = regi
31     }

```

L'erreur est que « cin » est dans une boucle et que par définition cette fonction ne consomme pas les délimiteurs. Cette erreur a été localisée en survolant le code à la recherche d'erreur de syntaxe. (C'était une mauvaise pratique)

Solution

```
25     for (i = 0; i < MAX_CAS; ++i)
26     {
27         for (int j = 0; j < MAX_VALE
28         {
29             cin >> registre;
30             cin.ignore();
31             tabValeurs[i, j] = regis
32     }
```

Erreur 6

Localisation

programmePrincipal.cpp

```
24     int registre;
44     float taux;
45     int cardinalite = MAX_VALEURS * MAX_CAS;
```

Les erreurs précédentes sont des déclarations de variables « local » au main. La déclaration de variable devrait ce faire au début de la fonction. Ces erreurs ont été localisées en survolant le code à la recherche d'erreur de syntaxe.

Solution

```
17 int main()
18 {
19     float variante;
20     int tabValeurs[MAX_CAS][MAX_VAL
21     int registre;
22     float taux;
```

Erreur 7

Localisation

programmePrincipal.cpp

```
26     for (i = 0; i < MAX_CAS; ++i)
```

C'est une erreur de syntaxe, puisque « i » n'a pas été déclaré préalablement il faut donc déclarer « i » dans l'énoncé « for ». Cette erreur a été localisé en survolant le code à la recherche d'erreur de syntaxe.

Solution

```
26     for (int i = 0; i < MAX_CAS; ++i)
```

Erreur 8

Localisation

programmePrincipal.cpp

```
49     cin >> seuil;
```

Cette erreur a été localisé en survolant le code à la recherche d'erreur de syntaxe. Puisque dans les erreurs découvert préalablement toutes les variables du programme principal ont été mis au début de la fonction main() il a été facile de noter que seuil n'avait pas été déclaré. (C'était une erreur de logique)

Solution

```
14
15 int main()
16 {
17     float variante;
18     int tabValeurs[MAX_CAS][MAX_VAL
19     int registre;
20     float taux;
```

Erreur 9

Localisation

programmePrincipal.cpp

```
40 afficherTableau(tabValeurs, MAX_CAS);
```

On remarque un indicateur d'erreur sur eclipse. Puisque cette fonction n'est pas défini dans programme principal, on peut assumer qu'elle provient de fonctionUtilitaire.h.

```
2 * \file fonctionsUtilitaires.h
8 #ifndef FONCTIONSUTILITAIRES_H
9 #define FONCTIONSUTILITAIRES_H
10
11 const int MAX_CAS = 4;
12 const int MAX_VALEURS = 4;
13
14
15 void triBulle(int tabDonnees[MAX_VALEURS]);
16 void tri2d(int tabDonnees[][MAX_VALEURS], int);
17 int occurrencesPlusGrand(int tabDonnees[MAX_VALEURS]
18 bool existe(int tabDonnees[][MAX_VALEURS], int, int)
```

La fonction afficherTableau n'est pas présente dans le fichier « .h » on va donc investiguer le fichier « .cpp »

```
82 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int p_nombreLignes)
```

On trouve donc la fonction dans le « .cpp » il faut donc mettre une référence à cette fonction dans le « .h ». (C'était une erreur d'édition de lien)

Solution

fonctionsUtilitaires.h

```
19 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int p_nombreLignes);
```

Erreur 10

Localisation

fonctionsUtilitaires.cpp

```
82 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int
83 {
84     for (int i = 0; i < p_nombreLignes; i++)
```

Dans la dernière erreur on a découvert la fonction afficherTableau dans le document fonctionsUtilitaires.cpp. On remarque la puce qui indique une possible erreur. On note que dans la

déclaration de la boucle « for » l'opérateur de comparaison n'est pas utilisé correctement. (C'était une erreur de syntaxe)

Solution

```
82 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int  
83 {  
84     for (int i = 0; i < p_tabDonnees[0].taille; i++)
```

Erreur 11

Localisation

fonctionsUtilitaire.cpp

```
43 void tri2d(int tabDonnees[][MAX_VALEURS], int p_nomb  
44 {  
45     for (int i = 0; i < p_nombre-de-Lignes; i++)  
46     {  
47         triBulle(tabDonnees[i]);  
48     }
```

En survolant le code à la recherche d'erreur de syntaxe, on remarque à la fonction « tri2d » une pastille rouge qui indique une erreur de syntaxe. On note la déclaration du paramètre p_nombre-nombre-Lignes n'est pas fait correctement puisqu'on ne peut pas séparer les mots d'une variable avec des traits d'union. (C'était une erreur de syntaxe)

Solution

fonctionUtilitaire.cpp

```
43 void tri2d(int tabDonnees[][MAX_VALEURS], int p_nomb  
44 {  
45     for (int i = 0; i < p_nombreDeLignes; i++)  
46     {  
47         triBulle(tabDonnees[i]);  
48     }
```

Erreur 12

Localisation

fonctionsUtilitaires.cpp

```
17 void triBulle(int tabDonnee[MAX_  
18 {  
19     int i, j, flag = 1;
```

En survolant la fonction « triBulle » à la recherche d'erreur de syntaxe, on remarque tout de suite une mauvaise pratique dans la déclaration de variable bien que celle-ci soit valide. « i » et « j » n'ont pas besoin d'être déclarés puisqu'on peut les déclarer dans la déclaration de la boucle « for ».

Solution

fonctionUtilitaires.cpp


```

17 void triBulle(int tabDonnee[MAX_VALEURS])
18 {
19     int flag = 1;
20     int temp;
21
22     for (int i = 1; (i <= MAX_VALEURS) && flag; i++)
23     {
24         flag = 0;
25         for (int j = 0; j < (MAX_VALEURS - 1); j++)
26         {
27             if (tabDonnee[j] > tabDonnee[j + 1])
28             {
29                 temp = tabDonnee[j];
30                 tabDonnee[j] = tabDonnee[j + 1];
31                 tabDonnee[j + 1] = temp;
32                 flag = 1;
33             }
34         }
35     }
36 }

```

Erreur 13

Localisation

fonctionsUtilitaires.h

```

15 void triBulle(int tabDonnees[MAX_VALEURS]);
16 void tri2d(int tabDonnees[][MAX_VALEURS], int);
17 int occurrencesPlusGrand(int p tabDonnees[MAX_VALEURS][], int);
18 bool existe(int tabDonnees[][MAX_VALEURS], int, int p_valeur);

```

En survolant le document « .h » du groupe fonctionsUtilitaires à la recherche d'erreur, on remarque qu'il y a plusieurs paramètres des fonctions qui n'ont pas de nom. Malgré qu'il soit légal de ne pas donner de nom à c'est paramètre c'est une très mauvaise pratique. On peut donc aller dans le « .cpp » afin de compléter les trous. On remarque que la fonction « existe » n'existe pas mais qu'elle est employée dans la fonction occurrencesPlusGrand. (C'était une mauvaise pratique)

Solution

```

14
15 void triBulle(int tabDonnees[MAX_VALEURS]);
16 void tri2d(int tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreDeLignes);
17 int occurrencesPlusGrand(int p_tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreDeLignes);
18 bool existe(int tabDonnees[MAX_CAS][MAX_VALEURS], int p_valeur);

```

Erreur 14

Localisation

fonctionsUtilitaires.h

```

15 void triBulle(int tabDonnees[MAX_VALEURS]);
16 void tri2d(int tabDonnees[][MAX_VALEURS], int p_nombreDeLignes);
17 int occurrencesPlusGrand(int p_tabDonnees[MAX_VALEURS][], int p_nombreDeLignes);
18 bool existe(int tabDonnees[][MAX_VALEURS], int, int p_valeur);

```

En corrigeant l'erreur précédente, on peut remarquer que la constante MAX_VALEURS est utilisée pour donner une dimension au tableau « tabDonnees[][] » mais qu'elle n'est pas utilisée pour définir la même dimension. On cherche donc dans le programme principal comment les dimensions sont définies.

```
int tabValeurs[MAX_CAS][MAX_VALEURS];
```

Solution

```
15 void triBulle(int tabDonnees[MAX_VALEURS]);
16 void tri2d(int tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreDeLignes);
17 int occurrencesPlusGrand(int p_tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreLi
18 bool existe(int tabDonnees[MAX_CAS][MAX_VALEURS], int, int p_valeur);
```

Erreur 15

Localisation

fonctionUtilitaires.cpp

```
84 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int
85 {
86     for (int i = 0; i == p_nombreLignes; i++)
87     {
88         cout << "Ligne " << i + 1 << " : ";
89         for (int j = 0; j == MAX_VALEURS; j++)
90         {
91             cout << " " << p_tabDonnees[i][j];
92         }
93     }
94 }
```

En survolant le code à la recherche d'erreur de syntaxe on remarque dans la fonction « afficherTableau » qu'il n'y a pas de « endl ». (C'était une erreur de syntaxe)

Solution

```
85 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int
86 {
87     for (int i = 0; i == p_nombreLignes; i++)
88     {
89         cout << "Ligne " << i + 1 << " : " << endl;
90         for (int j = 0; j == MAX_VALEURS; j++)
91         {
92             cout << " " << p_tabDonnees[i][j] << endl;
93         }
94     }
95 }
```

Erreur 16

Localisation

fonctionUtilitaires.cpp

```
8 #include <istream>
9 #include "fonctionsUtilitaires.h"
```

En survolant « fonctionsUtilitaires » on remarque que certaines fonctions contiennent des « cout » mais au début du document « istream » n'est pas inclus. (C'était une erreur d'édition de lien)

Solution

```
8 #include <istream>
9 #include "fonctionsUtilitaires.h"
10 #include <iostream>
```

Erreur 17

Localisation

```
programmePrincipal.cpp
27         for (int i = 0; i < MAX_CAS; ++i)
28         {
29             for (int j = 0; j < MAX_VALEURS; ++j)
30             {
31                 cin >> registre;
32                 cin.ignore();
```

En survolant le code pour des erreurs de syntaxe on remarque la pastille rouge. On peut tout de suite comprendre que le tableau à 2 dimension n'est pas bien appelé. (C'était une erreur de syntaxe)

Solution

```
31                 cin >> registre;
32                 cin.ignore();
33                 tabValeurs[i][j] = registre;
```

Erreur 18

Localisation

```
programmePrincipal.cpp
40     afficherTableau(tabValeurs, MAX_CAS);
41
fonctionsUtilitaires.h
void afficherTableau(int p_tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreLignes);

fonctionsUtilitaires.cpp
85 void afficherTableau(int p_tabDonnees[][MAX_VALEURS], int
86 {
87     for (int i = 0; i == p_nombreLignes; i++)
88     {
89         cout << "Ligne " << i + 1 << " : " << endl;
90         for (int j = 0; j == MAX_VALEURS; j++)
91         {
92             cout << " " << p_tabDonnees[i][j] << endl ;
93         }
94     }
```

En lisant la fonction « afficherTableau », on peut clairement voir avec le savoir que nous possédons sur les tableaux que cette fonction pourrait seulement posséder 1 paramètre excluant la taille du tableau qui est forcé par les constante présentes dans le « .h ». De plus, on remarque que la condition pour « i » & et « j » n'est pas correct. (Mauvaise pratique et erreur de syntaxe)

Solution

```
fonctionsUtilitaires.h
19 void afficherTableau(int p_tabDonnees[MAX_CAS][MAX_VALEURS]);
fonctionsUtilitaires.cpp
85 void afficherTableau(int p_tabDonnees[MAX_CAS][
86 {
87     for (int i = 0; i < MAX_CAS; i++)
88     {
89         cout << "Ligne " << i + 1 << " : " << e
90         for (int j = 0; j < MAX_VALEURS; j++)
91         {
92             cout << p_tabDonnees[i][j] << endl
93         }
94     }
```

Erreur 19

Localisation

```
programmePrincipal.cpp
41
42     tri2d(tabValeurs, MA
fonctionsUtilitaires.h
16 void tri2d(int tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreDeLignes);
fonctionsUtilitaires.cpp
45 void tri2d(int tabDonnees[][MAX_VALEURS], int p_no
46 {
47     for (int i = 0; i < p_nombreDeLignes; i++)
48     {
49         triBulle(tabDonnees[i]);
50     }
```

En regardant la fonction tri2d, on comprend que cette fonction sert à utiliser une fonction de trie pour chaque colonne d'un tableau [colonne] [ligne]. On peut donc assumer que cette fonction prend seulement un tableau 2D en paramètre. C'était donc une erreur dans la logique de la fonction.

Solution

```
45 void tri2d(int tabDonnees[MAX_CAS][MA
46 {
47     for (int i = 0; i < MAX_CAS; i++)
48     {
49         triBulle(tabDonnees, i);
50     }
```

Erreur 20

Localisation

fonctionsUtilitaires.h

```

15 void triBulle(int tabDonnees[MAX_VALEURS]);
fonctionsUtilitaires.cpp
18 void triBulle(int tabDonnee[MAX_VALEURS])
19 {
20     int flag = 1;
21     int temp;
22
23     for (int i = 1; (i <= MAX_VALEURS) && flag; i++)
24     {
25         flag = 0;
26         for (int j = 0; j < (MAX_VALEURS - 1); j++)
27         {
28             if (tabDonnee[j + 1] > tabDonnee[j])
29             {
30                 temp = tabDonnee[j];
31                 tabDonnee[j] = tabDonnee[j + 1];
32                 tabDonnee[j + 1] = temp;
33                 flag = 1;
34             }
35         }
36     }
37 }

```

En regardant la fonction « triBulle », on réalise très vite que c'est un Bubble Sort. En observant attentivement l'algorithme on peut noter qu'il n'y a pas d'erreur dans celui-ci. La fonction Bubble sort dans l'état présent ne peut trier que des tableaux à une dimension. Cependant dans l'utilisation que le programme en fait il faut trier un tableau 2D on peut donc remarquer directement qu'il manque une dimension à notre tableau en paramètre ainsi qu'une référence à la colonne ou on est rendu. (voir erreur 19) L'erreur est donc encore une fois une erreur de logique donc la déclaration et l'utilisation des paramètres.

Solution

fonctionsUtilitaires.cpp

```

18 void triBulle(int tabDonnee[MAX_CAS][MAX_VALEURS], int p_ligne)
19 {
20     int flag = 1;
21     int temp;
22
23     for (int i = 1; (i <= MAX_VALEURS) && flag; i++)
24     {
25         flag = 0;
26         for (int j = 0; j < (MAX_VALEURS - 1); j++)
27         {
28             if (tabDonnee[p_ligne][j + 1] > tabDonnee[p_ligne][j])
29             {
30                 temp = tabDonnee[p_ligne][j];
31                 tabDonnee[p_ligne][j] = tabDonnee[p_ligne][j + 1];
32                 tabDonnee[p_ligne][j + 1] = temp;
33                 flag = 1;
34             }
35         }
36     }
37 }

```

Erreur 21

Localisation

```
fonctionsUtilitaires.h
18 bool existe(int tabDonnees[MAX_CAS][MAX_VALEURS], int p_valeur);
fonctionsUtilitaires.cpp
65 if (existe(p_tabDonnees, p_valeur))
66 {
```

Comme mentionné plus dans une erreur corrigée antérieurement, la fonction « existe » n'existe pas. En observant la fonction « occurrencesPlusGrand », on comprend que la fonction « existe » sert à déterminer si une valeur est présente dans le tableau. (C'est une erreur de logique ?)

Solution

```
fonctionsUtilitaires.cpp
102 bool existe(int tabDonnees[MAX_CAS][MAX_VALEURS], int
103             bool validation = false;
104
105     for (int i = 0; i < MAX_CAS; i++)
106     {
107         for (int j = 0; j < MAX_VALEURS; j++)
108         {
109             if (tabDonnees[i][j] == p_valeur){
110                 validation = true;
111                 break;
112             }
113         }
114     }
```

Erreur 22

Localisation

```
fonctionsUtilitaires.h
17 int occurrencesPlusGrand(int p_tabDonnees[MAX_CAS][MAX_VALEURS], int p_nombreLignes, int p_valeur);
fonctionsUtilitaires.cpp
62 int occurrencesPlusGrand(int p_tabDonnees[MAX_VALEURS][MAX_CAS], int p_nombreL
63 {
64     float nombreOccurrencesPlusGrand;
65     if (existe(p_tabDonnees, p_valeur))
66     {
67         for (int i = 0; i < p_nombreLignes; ++i)
68         {
69             int j = 0;
70             while (p_tabDonnees[i][j] > p_valeur || j > MAX_VALEURS)
71             {
72                 nombreOccurrencesPlusGrand++;
73                 j++;
74             }
75         }
```

En regardant la fonction « occurrencesPlusGrand », on constate qu'il y a une déclaration de variable inutile puisque le « p_nombreLignes » est équivalent à « MAX_CAS ». De plus, on remarque qu'il y a un « if » qui entoure les boucles qui entourent les boucles qui comptent la récurrence. Dans un cas où la valeur seuil n'est pas contenue dans le tableau 2D la fonction « occurrencesPlusGrand » devient complètement inutile. La logique derrière le « while » est aussi un peu douteuse. (Erreur de syntaxe, erreur de logique)

Solution

```
62 int occurrencesPlusGrand(int p_tabDonnees[MAX_CAS][MAX_VALEURS]
63 {
64     float nombreOccurrencesPlusGrand;
65
66     if (existe(p_tabDonnees, p_valeur))
67     {
68         cout<<p_valeur<<" est présent dans le tableau"<<endl;
69     }
70
71     for (int i = 0; i < MAX_CAS; ++i)
72     {
73         for (int j = 0; j < MAX_VALEURS; ++j)
74         {
75             if (p_tabDonnees[i][j] > p_valeur){
76                 nombreOccurrencesPlusGrand++;
77             }
78         }
79     }
```

Erreur 23

Localisation

programmePrincipal.cpp

```
40
41
42
43
44
45
46
47     taux = (occurrencesPlusGrand(tabValeurs, MAX_CAS, seuil) / ca
48     cout << "il y a " << taux << " pourcent de nombres plus gran
49         << seuil << " dans le tableau de valeurs" << endl;
```

Si on regarde l'utilisation de la fonction « occurrencesPlusGrand », on remarque tout de suite une mauvaise pratique. Il faudrait mettre les valeurs dans des variables avant de les utiliser. De plus, vu que « occurrencesPlusGrand » nous donne un « int » et que la division de 2 « int » donne généralement un « float » ou un « double » il faut s'assurer de garder les décimales lors de la division.

Solution

```
52
53     variante = occurrencesPlusGrand(tabValeurs, seuil);
54     taux = (variante / cardinalite)*100;
55
56     cout << "il y a " << taux << " pourcent de nombres pl
```

Conclusion

En conclusion, le code est fonctionnel malgré que les fonctions ne sont probablement pas optimisées au maximum. Du au peu de commentaire présent dans le code il est difficile de savoir ce qui est utile (ex : fonction existe) et de comprendre le fonctionnement de certain algorithme sans recherche approfondie sur internet (ex : Bubble Sort). Je comprends la pertinence de ce type d'exercice, mais il y avait beaucoup trop d'erreur de type mauvaise pratique pour la grosseur du travail. La quantité d'erreur « mauvaise pratique & syntaxe » un peu exorbitante rendait elle aussi la compréhension et correction de la logique trop difficile considérant les objectifs de ce travail.