

Nom	Prénom	Matricule	Programme
Gagné	Nicolas	111 124 001	B-Informatique

Travail Pratique 1

Rapport d'investigation:

Cahier des charges

Ce programme doit prendre en entrée un nombre d'étudiants choisis par l'utilisateur ainsi que leurs résultats d'examens. Le nombre d'étudiants choisis ne doit pas dépasser 4 et le nombre d'examen ne doit pas dépasser 2. Les données recueillies seront mémorisées dans un tableau pour lequel des fonctions auront été conçues afin de réaliser des calculs. Parmi ces calculs, le programme pourra afficher en sortie la taille de notre tableau, la note minimum obtenue, la note maximale obtenue ainsi que l'écart-type de notre échantillon de résultat.

La taille de notre tableau, selon les notes de cours, est en fait le nombre de cases remplies de celui-ci. Donc dans ce problème, étant donné que l'on prépare notre tableau d'avance avec des valeurs constantes, si on veut que le résultat soit dynamique en fonction des entrées de l'utilisateur, la taille sera équivalente au nombre d'examens multiplié par le nombre d'élèves entré.

La note minimum sera simplement la note la moins élevée parmi toutes nos observations.

La note maximum sera simplement la note la plus élevée parmi toutes nos observations.

Pour l'écart-type, voici la formule que nous allons utiliser, qui est celle de l'écart-type d'un échantillon :

Formula

$$s = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}}$$

s = sample standard deviation

N = the number of observations

x_i = the observed values of a sample item

\bar{x} = the mean value of the observations

Le résultat du programme principal ne donne qu'une seule écart-type en console, et non l'écart-type de chaque examen. Donc nous allons corriger le code donné pour calculer l'écart-type de tout l'échantillon.

Stratégie

L'outil de débogueur sera majoritairement utilisé afin de cerner les problèmes qui existent dans ce projet. Notamment, en essayant d'abord de compiler le programme, des erreurs seront trouvées directement dans notre console de débogage. Par la suite, on pourra effectuer des tests qui valideront si les résultats sont cohérents, s'ils ne le sont pas, l'outil de débogueur « étape par étape » sera utilisé pour cerner les erreurs de logique et d'exécution dans le code.

Erreur 1 : fichier programmePrincipal.cpp ligne 11-12

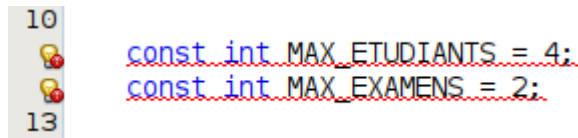
Localisation

```
g++-10 -c -g -MMD -MP -MF "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d" -o build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o ../programmePrincipal.cpp
../programmePrincipal.cpp:11:11: error: redefinition of 'const int MAX_ETUDIANTS'
11 | const int MAX_ETUDIANTS = 4;
    | ^~~~~~
In file included from ../programmePrincipal.cpp:8:
../fonctionsUtilitaires.h:14:11: note: 'const int MAX_ETUDIANTS' previously defined here
14 | const int MAX_ETUDIANTS = 4;
    | ^~~~~~
../programmePrincipal.cpp:12:11: error: redefinition of 'const int MAX_EXAMENS'
12 | const int MAX_EXAMENS = 2;
    | ^~~~~~
In file included from ../programmePrincipal.cpp:8:
../fonctionsUtilitaires.h:13:11: note: 'const int MAX_EXAMENS' previously defined here
13 | const int MAX_EXAMENS = 2;
    | ^~~~~~
```

Ici, les lignes : « ../programmePrincipal.cpp:11:11: error: redefinition of 'const int MAX_ETUDIANTS' » et « ../programmePrincipal.cpp:12:11: error: redefinition of 'const int MAX_EXAMENS' » précisent les erreurs des lignes 11-12.

Justification:

programmePrincipal.cpp lignes 11-12



```
10
11 | const int MAX_ETUDIANTS = 4;
12 | const int MAX_EXAMENS = 2;
13 | const int MAX_EXAMENS = 2;
```

Explications :

Au début du programme principal, on déclare les deux constantes MAX_ETUDIANTS et MAX_EXAMENS alors que le compilateur a vu que ces deux variables ont bien été déclarées déjà dans le fichier fonctionsUtilitaires.h préalablement compilé durant la pré-compilation.

Solution

Supprimer le contenu actuel des lignes 11-12 du fichier programmePrincipal.cpp .

```
6  #include <iostream>
7  #include <array>
8  #include "fonctionsUtilitaires.h"
9  using namespace std;
10
11  |
12  int
13  main ()
14  {
```

Justification:

```
cd '/home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation'
/usr/bin/make -f Makefile CONF=Debug
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
make[1] : on entre dans le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
"/usr/bin/make" -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux/investigation
make[2] : on entre dans le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
mkdir -p build/Debug/GNU-Linux/_ext/Sc0
rm -f "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d"
g++-10 -c -g -MMD -MP -MF "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d" -o build/Debug/GNU-Linux/_ext/Sc0/progra
../programmePrincipal.cpp: In function 'int main()':
../programmePrincipal.cpp:37:3: error: 'afficherTableau' was not declared in this scope
   37 |   afficherTableau (tabNotes, &nbEleves);
      |   ^~~~~~
make[2]: *** [nbproject/Makefile-Debug.mk:74 : build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o] Erreur 1
make[2] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make[1]: *** [nbproject/Makefile-Debug.mk:60 : .build-conf] Erreur 2
make[1] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make: *** [nbproject/Makefile-impl.mk:40 : .build-impl] Erreur 2

BUILD FAILED (exit value 2, total time: 574ms)
```

Explications :

On voit grâce à la capture d'écran en « justification » que les messages d'erreurs concernant les lignes 11-12 sont bel et bien disparus.

Erreur 2 : fichier programmePrincipal.cpp ligne 37

Localisation

```
g++-10 -c -g -MMD -MP -MF "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d" -o build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o ../programmePrincipal.cpp
../programmePrincipal.cpp: In function 'int main()':
../programmePrincipal.cpp:37:3: error: 'afficherTableau' was not declared in this scope
   37 |   afficherTableau (tabNotes, &nbEleves);
```

La ligne bleue de cette capture d'écran montre bien l'erreur de compilation qui est à la ligne 37.

programmePrincipal.cpp ligne 37

```
38  |   afficherTableau (tabNotes, &nbEleves);
39  |   return 0;
40  | }
```

On nous dit que la fonction afficherTableau n'a pas été déclarée dans la portée du programme principal. La fonction a bel et bien été créée dans le fichier fonctionsUtilitaires.cpp, cependant le prototype de la fonction n'a pas été ajouté au fichier fonctionsUtilitaires.h pour s'assurer que la fonction est disponible pour être utilisée dans notre programme principal. Cela nous donne une erreur d'édition de liens. La capture d'écran ci-bas montre bien que le prototype de la fonction afficherTableau n'a pas été déclarée :

```

□ #ifndef FONCTIONSUTILITAIRES_H_
#define FONCTIONSUTILITAIRES_H_
#include <array>

const int MAX_EXAMENS = 2;
const int MAX_ETUDIANTS = 4;

void saisieNotes (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS > p_tabNotes, int*);
int minimum (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int*);
int maximum (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int*);
double ecartType (std::array<int, MAX_EXAMENS>& p_tabNotesEtudiant);

#endif /* FONCTIONSUTILITAIRES_H_ */

```

Solution

On ajoute simplement le prototype de la fonction afficherTableau dans le fichierUtilitaires.h comme suit :

```

□ #ifndef FONCTIONSUTILITAIRES_H_
#define FONCTIONSUTILITAIRES_H_
#include <array>

const int MAX_EXAMENS = 2;
const int MAX_ETUDIANTS = 4;

void saisieNotes (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS > p_tabNotes, int*);
int minimum (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int*);
int maximum (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int*);
double ecartType (std::array<int, MAX_EXAMENS>& p_tabNotesEtudiant);
void afficherTableau (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbElevés);

#endif /* FONCTIONSUTILITAIRES_H_ */

```

Voilà, l'erreur est maintenant disparue (capture d'écran ci-bas). Le compilateur reconnaît maintenant toutes les fonctions que l'on a utilisé dans notre programme principal.

```

g++10 -c -g -MMD -MP -MF "build/Debug/GNU-Linux/_ext/Sc0/fonctionsUtilitaires.o.d" -o build/Debug/GNU-Linux/_ext/Sc0/fonctionsUtilitaires.o ../fonctionsUtilitaires.cpp
../fonctionsUtilitaires.cpp: In function 'int minimum(std::array<std::array<int, 2>, 4>&, int*)':
../fonctionsUtilitaires.cpp:43:1: warning: no return statement in function returning non-void [-Wreturn-type]
  43 | }
     | ^
mkdir -p build/Debug/GNU-Linux/_ext/Sc0
rm -f "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d"
g++10 -c -g -MMD -MP -MF "build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o.d" -o build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o ../programmePrincipal.cpp
mkdir -p dist/Debug/GNU-Linux
g++10 -o dist/Debug/GNU-Linux/investigation build/Debug/GNU-Linux/_ext/Sc0/fonctionsUtilitaires.o build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o
/usr/bin/ld : build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o : dans la fonction « main » :
/home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation/.../programmePrincipal.cpp:31 : référence indéfinie vers « saisieNotes(std::array<std::array<int, 2>
collect2: error: ld returned 1 exit status
make[2]: *** [nbproject/Makefile-Debug.mk:64 : dist/Debug/GNU-Linux/investigation] Erreur 1
make[2] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make[1]: *** [nbproject/Makefile-Debug.mk:60 : .build-conf] Erreur 2
make[1] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make: *** [nbproject/Makefile-impl.mk:40 : .build-impl] Erreur 2

BUILD FAILED (exit value 2, total time: 1s)

```

Cependant, on remarque qu'une nouvelle erreur de compilation est apparue qu'on n'avait pas auparavant. Continuons avec cette erreur pour bien l'expliquer...

Erreur 3 : fichier fonctionsUtilitaires.cpp ligne 43

Localisation

```
g++10 -c -g -MD -MP -MF "build/Debug/GNU-Linux_ext/5c0/fonctionsUtilitaires.o.d" -o build/Debug/GNU-Linux_ext/5c0/fonctionsUtilitaires.o ../fonctionsUtilitaires.cpp
../fonctionsUtilitaires.cpp: In function 'int minimum(std::array<std::array<int, 2>, 4>&, int*)':
../fonctionsUtilitaires.cpp:43:1: warning: no return statement in function returning non-void [-Wreturn-type]
  43 | }
      | ^
mkdir -p build/Debug/GNU-Linux_ext/5c0
rm -f build/Debug/GNU-Linux_ext/5c0/programmePrincipal.o.d
g++10 -c -g -MD -MP -MF "build/Debug/GNU-Linux_ext/5c0/programmePrincipal.o.d" -o build/Debug/GNU-Linux_ext/5c0/programmePrincipal.o ../programmePrincipal.cpp
mkdir -p dist/Debug/GNU-Linux
g++10 -o dist/Debug/GNU-Linux/investigation build/Debug/GNU-Linux_ext/5c0/fonctionsUtilitaires.o build/Debug/GNU-Linux_ext/5c0/programmePrincipal.o
/usr/bin/ld : build/Debug/GNU-Linux_ext/5c0/programmePrincipal.o : dans la fonction « main » :
/home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation/.../programmePrincipal.cpp:31 : référence indéfinie vers « saisieNotes(std::array<std::array<int, 2>
collect2: error: ld returned 1 exit status
make[2]: *** [nbproject/Makefile-Debug.mk:64 : dist/Debug/GNU-Linux/investigation] Erreur 1
make[2]: on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make[1]: *** [nbproject/Makefile-Debug.mk:60 : .build-conf] Erreur 2
make[1]: on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make: *** [nbproject/Makefile-impl.mk:40 : .build-impl] Erreur 2

BUILD FAILED (exit value 2, total time: 1s)
```

La ligne bleue de la capture précédente nous confirme que l'erreur est localisée à la ligne 43 du fichier fonctionsUtilitaires.cpp que voici :

fonctionsUtilitaires.cpp ligne 43

```
29 int minimum (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbElevés)
30 {
31     int noteFaible = 100;
32     for (int i = 0; i < MAX_ETUDIANTS; i++)
33     {
34         for (int j = 0; j < MAX_EXAMENS; j++)
35         {
36             if (p_tabNotes[i][j] < noteFaible)
37             {
38                 noteFaible = p_tabNotes[i][j];
39             }
40         }
41     }
42 }
43 }
```

On voit très bien que le prototype de la fonction minimum à la ligne 29 précise le retour d'un entier, qui n'est jamais retourné à l'intérieur des surlignements jaunes.

Solution

Il faut ajouter une valeur de retour. Ici, lors de l'itération au travers des données, on stocke la note la plus faible dans la variable noteFaible. Donc lorsque l'itération complète est terminée, c'est bien cette valeur que nous souhaitons retourner. Donc on devra ajouter la ligne « return noteFaible; » à l'intérieur des surlignements jaunes, mais après la grande boucle « pour tout » comme suit :

```
30 int
31 minimum (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbElevés)
32 {
33     int noteFaible = 100;
34     for (int i = 0; i < MAX_ETUDIANTS; i++)
35     {
36         for (int j = 0; j < MAX_EXAMENS; j++)
37         {
38             if (p_tabNotes[i][j] < noteFaible)
39             {
40                 noteFaible = p_tabNotes[i][j];
41             }
42         }
43     }
44     return noteFaible;
45 }
```

L'erreur a bien été corrigée. Il n'y a plus de ligne bleue dans notre résultat de compilation :

```
g++-10 -o dist/Debug/GNU-Linux/investigation build/Debug/GNU-Linux/_ext/Sc0/fonctionsUtilitaires.o build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o
/usr/bin/ld : build/Debug/GNU-Linux/_ext/Sc0/programmePrincipal.o : dans la fonction « main » :
/home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation/./programmePrincipal.cpp:31 : référence indéfinie vers « saisieNotes(std::array<std::array<int, 2u> collect2: error: ld returned 1 exit status
make[2]: *** [nbproject/Makefile-Debug.mk:64 : dist/Debug/GNU-Linux/investigation] Erreur 1
make[2] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make[1]: *** [nbproject/Makefile-Debug.mk:60 : .build-conf] Erreur 2
make[1] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make: *** [nbproject/Makefile-impl.mk:40 : .build-impl] Erreur 2

BUILD FAILED (exit value 2, total time: 578ms)
```

Erreur 4 : fichier programmePrincipal.cpp ligne 31

Localisation

Si on grossit sur la ligne qui localise l'erreur à partir de la dernière capture d'écran ci-haut :

```
/home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation/./programmePrincipal.cpp:31 : référence indéfinie vers « saisieNotes(std::array<std::array<int, 2u> collect2: error: ld returned 1 exit status
```

On voit bien ici que l'erreur se trouve à la ligne 31 du fichier programmePrincipal.cpp .

programmePrincipal.cpp ligne 31

```
31 | | | saisieNotes (tabNotes, &nbElevés);
    | | | cout << "taille" << tabNotes.size () << endl;
```

La ligne 31 contient l'appel à la fonction `saisieNotes` avec les paramètres demandés, dont une référence à l'emplacement en mémoire de la variable `nbElevés` (`&nbElevés`). Dans le message d'erreur fourni par le compilateur, cela nous indique une « référence indéfinie vers « `saisieNotes` » qui cause une erreur de sortie en fournissant la valeur de retour 1. Si on va dans le fichier `fonctionsUtilitaires.cpp` pour analyser la conception de la fonction `saisieNotes`, on aperçoit rapidement que le pointeur (`int* p_nbElevés`) ajouté dans les paramètres de la fonction lors de l'appel est déclaré à l'intérieur de la fonction `saisieNotes`, et non dans le prototype de celle-ci :

```
14 | void
15 | saisieNotes (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes)
16 | {
17 | | int* p_nbElevés;
```

Ce n'est pas le seul endroit où il y a une erreur de référence. Si on regarde bien dans le prototype de la fonction `saisieNotes` qui est déclaré dans le fichier `fonctionsUtilitaires.h`, on aperçoit également qu'il manque le caractère « `&` » qui indique qu'on va stocker une adresse comme référence :

```
16 | void saisieNotes (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS > p_tabNotes, int*);
```

Solution

Pour corriger la situation, on commencera simplement par déplacer la déclaration du pointeur `*p_nbElevés` à l'intérieur du prototype de la fonction `saisieNotes` dans le fichier `fonctionsUtilitaires.cpp` comme suit :

```
14 | void
15 | saisieNotes (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbElevés)
16 | {
17 | | for (int i = 0; i < *p_nbElevés; i++)
18 | | {
19 | | | for (int j = 0; j < MAX_EXAMENS; j++)
```

Ensuite, on va prendre soin d'inclure le caractère « `&` » juste avant notre identificateur du pointeur vers notre tableau de donnée `p_tabNotes` dans notre prototype de la fonction `saisieNotes` qui est déclaré dans le fichier `fonctionsUtilitaires.h` comme suit :

```
16 | void saisieNotes (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int*);
```

En analysant correctement le prototype de la fonction saisieNotes que l'on trouve dans les fichiers fonctionsUtilitaires.h et fonctionsUtilitaires.cpp , on voit qu'ils sont désormais cohérents dans leur déclaration de variables.

La compilation du code corrigé résout bel et bien cette erreur de référence en nous affichant un message « BUILD SUCCESSFUL » et la console nous demande la première entrée pour réaliser notre programme.

```
g++-10 -o dist/Debug/GNU-Linux/investigation build/Debug/GNU-Linux/_ext/5c0/fonctionsUtilitaires.o build/Debug/GNU-Linux/_ext/5c0/programmePrincipal.o
make[2] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
make[1] : on quitte le répertoire « /home/etudiant/gif_1003/travaux_pratiques/tpl/investigationErreurs/investigation »
```

```
BUILD SUCCESSFUL (total time: 675ms)
```

```
Saisissez le nombre d'élèves
```

```
█
```

Erreur 5 : fichier programmePrincipal.cpp lignes 32-33 et fichierUtilitaires.cpp lignes 33

Localisation

```
Saisissez le nombre d'élèves
```

```
1
```

```
Saisir uniquement des nombres entiers positifs et inférieurs à 100  
(ce programme ne prend pas en charge la validation de saisie)
```

```
Saisissez la note de l'examen 1
```

```
80
```

```
Saisissez la note de l'examen 2
```

```
85
```

```
taille 4
```

```
RUN FINISHED; Segmentation fault; core dumped; real time: 3s; user: 0ms; system: 0ms
```

```
█
```

Ici après avoir « testé » des valeurs, j'obtiens une erreur de segmentation après que la console a réussi à afficher la taille de notre tableau. Voici le résultat en donnant d'autres valeurs :

Investigation (Build)	Investigation (Clean, Build)	Investigation (Build, Run)	Investigation (Run)	Testvall
Saisissez le nombre d'élèves				
3				
Saisir uniquement des nombres entiers positifs et inférieurs à 100 (ce programme ne prend pas en charge la validation de saisie)				
Saisissez la note de l'examen 1				
80				
Saisissez la note de l'examen 2				
80				
Saisissez la note de l'examen 1				
80				
Saisissez la note de l'examen 2				
80				
Saisissez la note de l'examen 1				
80				
Saisissez la note de l'examen 2				
80				
taille 4				
RUN FINISHED; Segmentation fault; core dumped; real time: 8s; user: 0ms; system: 0ms				

La taille en résultat est la même. Donc forcément la « segmentation fault » indique une erreur de pointeur dans ces lignes et le fait que la taille est identique alors que mon nombre d'étudiants a changé indique une erreur de logique.

Voici les lignes 32 à 35 du fichier programmePrincipal.cpp :

```

32 cout << "taille " << tabNotes.size() << endl;
33 cout << "note minimum : " << minimum(tabNotes, &nbEleves) << endl;
34 cout << "note maximum : " << maximum(tabNotes, &nbEleves) << endl;
35 cout << "écart type : " << ecartType(tabNotes[0]) << endl;

```

Solution

Pour résoudre le problème de la taille pour débiter, affichons la bonne taille en modifiant la ligne 32 et en inscrivant « nbEleves * MAX_EXAMENS » à la place de « tabNotes.size() » comme suit :

```

cout << "taille " << nbEleves * MAX_EXAMENS << endl;

```


Ainsi, je fais le test avec 4 étudiants et j'obtiens une taille de 8 au lieu de 4 :

```
Saisissez le nombre d'élèves
4
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
90
Saisissez la note de l'examen 2
95
Saisissez la note de l'examen 1
85
Saisissez la note de l'examen 2
90
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
85
Saisissez la note de l'examen 1
50
Saisissez la note de l'examen 2
60
taille 8
```

Par la suite, pour le problème de pointeur, allons dans notre fichierUtilitaires.cpp dans la fonction minimum pour détecter le problème, étant donné que notre « segmentation fault » survient immédiatement après afficher la taille de notre tableau. Donc cela indique forcément que l'erreur se produit à la ligne 33 du fichier programmePrincipal.cpp, où l'utilisation de pointeurs se fait dans la fonction minimum. Dans la fonction minimum, on s'aperçoit que la 2^e boucle incrémente i et non j, donc nous devons corriger cela car on sort de notre espace de tableau initialisé en incrémentant seulement i. Par la même occasion, je vais optimiser la condition de la première boucle de cette fonction minimum pour qu'elle boucle seulement pour le nombre d'élèves que l'utilisateur aura entré, et non la valeur maximale prédéfinie. Voici le résultat de la correction de notre fonction minimum:

```
29  int
30  minimum (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbEleves)
31  {
32      int noteFaible = 100;
33      for (int i = 0; i < *p_nbEleves; i++)
34      {
35          for (int j = 0; j < MAX_EXAMENS; j++)
36          {
37              if (p_tabNotes[i][j] < noteFaible)
38              {
39                  noteFaible = p_tabNotes[i][j];
40              }
41          }
42      }
43      return noteFaible;
44  }
```

Ce qui nous donne maintenant un résultat en console plus exhaustif et on voit que notre programme principal s'est réalisé au complet avec la sortie « exit value 0 » en vert :

```
Saisissez le nombre d'élèves
1
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
95
Saisissez la note de l'examen 2
90
taille 1
note minimum : 90
note maximum :90
écart type : -nan
Notes de l'étudiant 1
Examen 1 : 95
Examen 2 : 90

RUN FINISHED; exit value 0; real time: 4s; user: 0ms; system: 0ms
```

□

Erreur 5 : fichier fonctionsUtilitaires.cpp lignes 50 et 55

Localisation

Maintenant que notre programmePrincipal.cpp fonctionne d'un bout à l'autre, nous devons régler les erreurs de logique de notre programme pour s'assurer que nous obtenons les bons résultats.

J'effectue donc ce premier test :

```
Saisissez le nombre d'élèves
2
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
90
Saisissez la note de l'examen 2
95
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
85
taille 2
note minimum : 80
note maximum :100
écart type : -nan
Notes de l'étudiant 1
Examen 1 : 90
Examen 2 : 95
Notes de l'étudiant 2
Examen 1 : 80
Examen 2 : 85
```

```
RUN FINISHED; exit value 0; real time: 6s; user: 0ms; system: 0ms
```

```
/home/etudiant/gif_1009/travaux_pratiques/tpl/investigationErreurs/investigation/./programmePrincipal.cpp:31 : référence indéfinie vers « saisieNotes(std::array<std::array<int, 2u/collect2: error: ld returned 1 exit status
```

Si je vais de haut en bas, je commence par remarquer que le maximum est erroné. Il devrait être à 95 et non 100. Si je vais dans le fichier fonctionsUtilitaires.cpp, je retrouve la fonction maximum qui contient 2 erreurs aux lignes surlignées en rouge :

```
47 int
48 maximum (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbEleves)
49 {
50     int noteeelevee = 100;
51     for (int i = 0; i < *p_nbEleves; i++)
52     {
53         for (int j = 0; j < MAX_EXAMENS; j++)
54         {
55             if (p_tabNotes[j][j] > noteeelevee)
56             {
57                 noteeelevee = p_tabNotes[i][j];
58             }
59         }
60     }
61     return noteeelevee;
62 }
63 }
```

À la ligne 50, on définit la variable noteeelevee directement avec la valeur 100, donc considérant que 100 est la note la plus élevée possible, cette valeur ne changera jamais.

À la ligne 55, la condition compare l'itération [j][j], où j sera soit 0 ou 1, donc la boucle « pour » tout testera seulement les cas du tableau [0][0] et [1][1].

Solution

Pour résoudre ce problème, on commence par modifier l'initialisation de la variable noteeelevee à l'intérieur de la fonction maximum (fichier fonctionsUtilitaires.cpp) pour l'initialiser à la valeur 0 au lieu de 100. Par la suite, on change la condition de la boucle « si » pour (p_tabNotes [i][j] > noteeelevee) comme suit :

```
47 int
48 maximum (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbEleves)
49 {
50     int noteeelevee = 0;
51     for (int i = 0; i < *p_nbEleves; i++)
52     {
53         for (int j = 0; j < MAX_EXAMENS; j++)
54         {
55             if (p_tabNotes[i][j] > noteeelevee)
56             {
57                 noteeelevee = p_tabNotes[i][j];
58             }
59         }
60     }
61     return noteeelevee;
62 }
63 }
```

Si ensuite je teste avec des valeurs en entrée, j'aurai maintenant le bon résultat de la fonction maximum en console comme suit :

```
> Saisissez le nombre d'élèves
2
> Saisir uniquement des nombres entiers positifs et inférieurs à 100
  (ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
90
Saisissez la note de l'examen 2
95
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
85
taille 2
note minimum : 80
note maximum : 95
écart type : -nan
Notes de l'étudiant 1
Examen 1 : 90
Examen 2 : 95
Notes de l'étudiant 2
Examen 1 : 80
Examen 2 : 85

RUN FINISHED; exit value 0; real time: 6s; user: 0ms; system: 0ms
```

Erreur 6 : fichier fonctionsUtilitaires.cpp lignes 66 à 85

Localisation

Si on poursuit avec l'analyse du dernier résultat en console, on remarque une valeur de -nan pour ce qui est de l'écart type des données. C'est forcément une erreur comme on aurait dû obtenir un écart-type de 6.45497224 selon Excel. Voici les lignes 66 à 85 du fichier fonctionsUtilitaires.cpp qui le corps de la fonction ecartType.

```
66     double
67     ecartType (array<int, MAX_EXAMENS>& p_tabNotesEtudiant)
68     {
69         int i = 0;
70         float somme = 0;
71         float sommeCarree = 0;
72
73         while (i < MAX_EXAMENS)
74         {
75             somme += p_tabNotesEtudiant[i];
76             sommeCarree = + p_tabNotesEtudiant[i] * p_tabNotesEtudiant[i];
77             i++;
78         }
79         float moyenne;
80         moyenne = somme / MAX_EXAMENS;
81         double variance;
82         variance = sommeCarree / MAX_EXAMENS - moyenne * moyenne;
83         double ecartType;
84         ecartType = sqrt (variance);
85         return ecartType;
86     }
```

Pour commencer, rappelons la formule de l'écart-type d'un échantillon :

Formula

$$s = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}}$$

s = sample standard deviation

N = the number of observations

x_i = the observed values of a sample item

\bar{x} = the mean value of the observations

Nous devons bien utiliser la formule pour un échantillon car nous traitons au plus 4 paires d'observations.

Plusieurs choses clochent dans le code de la fonction `ecartType`. Premièrement, pour la `sommeCarree` (qui représente en fait la somme des écarts avec la moyenne des observations élevée au carré), on voit que la valeur est définie à chaque fois par le carré de l'observation `i`, alors qu'on cherche à additionner la somme des écarts au carré. Ensuite, le calcul de la variance est erroné également, car la variance est le carré de l'écart-type, et pour obtenir la variance, on ne fait que diviser la somme des écarts au carré par notre nombre d'observations diminué de 1 ($N - 1$ car on traite un échantillon, et non une population complète). Dans le code précédent, pour obtenir la variance on divise la `sommeCarree` (qui est erronée) par le nombre maximal d'examen, tout cela soustrait ensuite par la moyenne (de seulement 1 étudiant, celui à l'indice 0 dans le tableau) au carré. Voici également la ligne de l'appel de la fonction `ecartType` (ligne 35 du fichier `programmePrincipal.cpp`) :

```
35 | cout << "écart type : " << ecartType (tabNotes[0]) << endl;
```

Solution

Je commence par corriger l'appel de la fonction pour parcourir toutes nos données recueillies dans le tableau à la ligne 35 du fichier `programmePrincipal.cpp` :

```
35 | cout << "écart type : " << ecartType (tabNotes, &nbEleves) << endl;
```

Étant donné que je fais ce changement, je dois également mettre à jour le prototype de la fonction `ecartType` dans les fichiers `fonctionsUtilitaires.h` :

```
19 | double ecartType (std::array<std::array<int, MAX_EXAMENS>, MAX_ETUDIANTS> &p_tabNotes, int*);
```

et `fonctionsUtilitaires.cpp` :

```
66 | double
67 | ecartType (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS> &p_tabNotes, int* p_nbEleves)
```

Par la suite, je définis une variable `nb_observations` à 0 pour compter mon nombre d'observations total à l'intérieur de ma fonction `ecartType` dans le fichier `fonctionsUtilitaires.cpp` :

```
69 | int nb_observations = 0;
```

Toujours dans le fichier `fonctionsUtilitaires.cpp`, je modifie la boucle `while` du code précédent à la ligne 73 pour une boucle « pour tout » similaire à celle de nos fonctions `minimum` et `maximum` pour bien obtenir toutes nos observations stockées en mémoire et calculer la somme de ces observations ainsi que le nombre d'observations pour être en mesure de calculer la moyenne de toutes les notes :

```
69 | int nb_observations = 0;
70 | float somme = 0;
71 | float sommeCarree = 0;
72 |
73 | // Première boucle pour calculer la somme totale des observations
74 | for (int i = 0; i < *p_nbEleves; i++)
75 | {
76 |     for (int j = 0; j < MAX_EXAMENS; j++)
77 |     {
78 |         somme += p_tabNotes[i][j];
79 |         nb_observations ++;
80 |     }
81 | }
82 | float moyenne;
83 | moyenne = somme / nb_observations;
```

Je fais ensuite une deuxième boucle « pour tout » très similaire à la première, mais cette fois pour calculer la somme des écarts au carré qui sera utile dans notre calcul pour trouver la variance :

```
85 // Deuxième boucle pour calculer la somme des écarts au carré
86 for (int i = 0; i < *p_nbEleves; i++)
87 {
88     for (int j = 0; j < MAX_EXAMENS; j++)
89     {
90         sommeCarree += pow ((p_tabNotes[i][j] - moyenne), 2);
91     }
92 }
```

Finalement, je corrige la formule de variance pour utiliser la bonne sommeCarree que je divise par le nombre d'observations calculé précédemment diminué de 1 (car on traite un échantillon).

```
94 variance = sommeCarree / (nb_observations - 1);
```

Voilà le résultat final de la fonction ecartType du fichier fonctionsUtilitaires.cpp (désormais lignes 66 à 98)

```
66 double
67 ecartType (array<array<int, MAX_EXAMENS>, MAX_ETUDIANTS >& p_tabNotes, int* p_nbEleves)
68 {
69     int nb_observations = 0;
70     float somme = 0;
71     float sommeCarree = 0;
72
73     // Première boucle pour calculer la somme totale des observations
74     for (int i = 0; i < *p_nbEleves; i++)
75     {
76         for (int j = 0; j < MAX_EXAMENS; j++)
77         {
78             somme += p_tabNotes[i][j];
79             nb_observations ++;
80         }
81     }
82     float moyenne;
83     moyenne = somme / nb_observations;
84
85     // Deuxième boucle pour calculer la somme des écarts au carré
86     for (int i = 0; i < *p_nbEleves; i++)
87     {
88         for (int j = 0; j < MAX_EXAMENS; j++)
89         {
90             sommeCarree += pow ((p_tabNotes[i][j] - moyenne), 2);
91         }
92     }
93     double variance;
94     variance = sommeCarree / (nb_observations - 1);
95     double ecartType;
96     ecartType = sqrt (variance);
97     return ecartType;
98 }
```


Première confirmation que le code fonctionne avec les entrées utilisées dans les captures précédentes :

```
Saisissez le nombre d'élèves
2
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
90
Saisissez la note de l'examen 2
95
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
85
taille 4
note minimum : 80
note maximum :95
écart type : 6.45497
Notes de l'étudiant 1
Examen 1 : 90
Examen 2 : 95
Notes de l'étudiant 2
Examen 1 : 80
Examen 2 : 85

RUN FINISHED; exit value 0; real time: 8s; user: 0ms; system: 0ms
```

Alors qu'Excel nous confirme les mêmes résultats que voici :

	Examen 1	Examen 2
Etudiant 1	90	95
Etudiant 2	80	85
Etudiant 3		
Etudiant 4		

Maximum	95
Minimum	80
Écart-Type	6.45497

Deuxième confirmation que le code fonctionne avec de nouvelles entrées :

```
Saisissez le nombre d'élèves
3
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
99
Saisissez la note de l'examen 1
80
Saisissez la note de l'examen 2
93
Saisissez la note de l'examen 1
40
Saisissez la note de l'examen 2
75
taille 6
note minimum : 40
note maximum :99
écart type : 20.6244
Notes de l'étudiant 1
Examen 1 : 80
Examen 2 : 99
Notes de l'étudiant 2
Examen 1 : 80
Examen 2 : 93
Notes de l'étudiant 3
Examen 1 : 40
Examen 2 : 75

RUN FINISHED; exit value 0; real time: 18s; user: 0ms; system: 0ms
□
```

Voici le résultat des mêmes données dans excel :

	Examen 1	Examen 2
Etudiant 1	80	99
Etudiant 2	80	93
Etudiant 3	40	75
Etudiant 4		

Maximum	99
Minimum	40
Écart-Type	20.6244

Troisième confirmation que le code fonctionne avec de nouvelles entrées :

```
Saisissez le nombre d'élèves
4
Saisir uniquement des nombres entiers positifs et inférieurs à 100
(ce programme ne prend pas en charge la validation de saisie)
Saisissez la note de l'examen 1
20
Saisissez la note de l'examen 2
99
Saisissez la note de l'examen 1
75
Saisissez la note de l'examen 2
94
Saisissez la note de l'examen 1
61
Saisissez la note de l'examen 2
85
Saisissez la note de l'examen 1
70
Saisissez la note de l'examen 2
95
taille 8
note minimum : 20
note maximum : 99
écart type : 25.8647
Notes de l'étudiant 1
Examen 1 : 20
Examen 2 : 99
Notes de l'étudiant 2
Examen 1 : 75
Examen 2 : 94
Notes de l'étudiant 3
Examen 1 : 61
Examen 2 : 85
Notes de l'étudiant 4
Examen 1 : 70
Examen 2 : 95

RUN FINISHED; exit value 0; real time: 15s; user: 0ms; system: 0ms
```

Voici le résultat des mêmes données dans excel :

	Examen 1	Examen 2
Etudiant 1	20	99
Etudiant 2	75	94
Etudiant 3	61	85
Etudiant 4	70	95

Maximum	99
Minimum	20
Écart-Type	25.8647

