

TP 1 – CODE REVIEW (DURÉE 2H)

Exercice 1 :

Question 1

Lancer le script « build.bat » dans le répertoire du TP1 et observer le résultat.

Le compilateur signale-t-il des alertes ? Si c'est le cas, corrigez-les avant de continuer.



```
C:\windows\system32\cmd.exe
X:\Semestre_8\MOQL\TP1>gcc -I. -c Main.c MatrixDrain.c SignCommand.c SignMatrix.c
X:\Semestre_8\MOQL\TP1>pause
Appuyez sur une touche pour continuer...
```

Il n'y a pas d'alerte.

Question 2

Nous allons maintenant demander au compilateur de réaliser une analyse plus approfondie du code lors de la compilation et de nous rapporter les problèmes rencontrés.

Pour cela, éditer le fichier « build.bat » et ajouter les options « -Wall -Wextra -Wpedantic » dans la ligne de compilation. Ceci active des alertes supplémentaires.

Relancer ensuite le script de compilation et observer le résultat.

Quelles alertes sont affichées ?

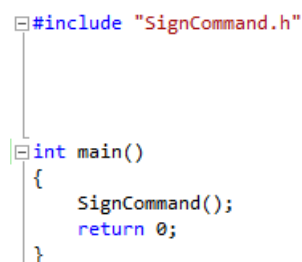
Corriger ces problèmes avant de continuer.

Les alertes affichées sont nombreuses, je vais les corriger dans l'ordre d'affichage.

1ere alerte :

```
X:\Semestre_8\MOQL\TP1>gcc -I. -c -Wall -Wextra -Wpedantic -Wmain Main.c MatrixDrain.c
SignCommand.c SignMatrix.c Main.c:6:9: warning: return type of 'main' is not 'int' [-Wmain]
UINT_32 main()
```

Dans le fichier main.c je modifie UINT_32 en int.



```
#include "SignCommand.h"

int main()
{
    SignCommand();
    return 0;
}
```

2^e alerte :

```
X:\Semestre_8\MOQL\TP1>gcc -I. -c -Wall -Wextra -Wpedantic Main.c MatrixDrain.c SignCommand.c SignMatrix.c
SignCommand.c: In function 'SpecifyArrow':
SignCommand.c:38:8: warning: format '%d' expects argument of type 'int *', but argument 2 has type 'enum Action *' [-Wformat=]
scanf("%d",&Choice);
      ^
```

Dans le fichier SignCommand.c, je change le %d de « choice » en %u car il caractérise les valeurs non signées :

```
void SpecifyArrow()
{
    enum Action Choice=Clear;
    static int MaxChoice=NW;
    int Count;

    printf(" \n");
    for (Count=0; Count<ARROWSCREENS; Count++)
    {
        printf("(%d) %s \n",Count,MatrixTexts[Count]);
    }
    scanf("%u",&Choice);
    printf ("\n");

    if (N<=Choice && MaxChoice>=Choice)
        UpdateDisplay(Choice);
}
```

3^e alerte :

```
X:\Semestre_8\MOQL\TP1>gcc -I. -c -Wall -Wextra -Wpedantic Main.c MatrixDrain.c SignCommand.c SignMatrix.c
SignCommand.c: In function 'SpecifyArrow':
SignCommand.c:41:28: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
    if (N<=Choice && MaxChoice>=Choice)
        ^
```

Dans le fichier SignCommand.c, j'ai forcé « choice » à être un entier avec « (int) »

```
void SpecifyArrow()
{
    enum Action Choice=Clear;
    static int MaxChoice=NW;
    int Count;

    printf(" \n");
    for (Count=0; Count<ARROWSCREENS; Count++)
    {
        printf("(%d) %s \n",Count,MatrixTexts[Count]);
    }
    scanf("%d",(int*)&Choice);
    printf ("\n");

    if (N<=(int)Choice && MaxChoice>=(int)Choice)
        UpdateDisplay(Choice);
}
```

4^e alerte :

```
X:\Semestre_8\MOQL\TP1>gcc -I. -c -Wall -Wextra -Wpedantic Main.c MatrixDrain.c SignCommand.c SignMatrix.c
In file included from SignMatrix.c:3:0:
Matrices.h:6:1: warning: missing braces around initializer [-Wmissing-braces]
{FALSE,FALSE,TRUE ,FALSE,FALSE,
```

Dans le fichier Matrices.h, je rajoute des accolades pour que les dimensions soient les bonnes :

```
#ifndef Matrices_H
#define Matrices_H

GuideMatrix ArrowStyleMatrix [SUBMATRIXTYPECOUNT]=
{
    //Straight
    {
        {FALSE,FALSE,TRUE ,FALSE,FALSE},\
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE},\
        {TRUE ,FALSE,TRUE ,FALSE,TRUE} ,\
        {FALSE,FALSE,TRUE ,FALSE,FALSE},\
        {FALSE,FALSE,TRUE ,FALSE,FALSE}
    },\
    //Angled
    {
        {TRUE ,TRUE ,TRUE ,TRUE ,FALSE},\
        {TRUE ,TRUE ,FALSE,FALSE,FALSE},\
        {TRUE ,FALSE,TRUE ,FALSE,FALSE},\
        {TRUE ,FALSE,FALSE,TRUE ,FALSE},\
        {FALSE,FALSE,FALSE,FALSE,TRUE}
    },\
    //Exclaim
    {
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE},\
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE},\
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE},\
        {FALSE,FALSE,FALSE,FALSE,FALSE},\
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE}
    },\
    //Stop
    {
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE},\
        {TRUE ,FALSE,FALSE,FALSE,TRUE} ,\
        {TRUE ,TRUE ,TRUE ,TRUE ,TRUE} ,\
        {TRUE ,FALSE,FALSE,FALSE,TRUE} ,\
        {FALSE,TRUE ,TRUE ,TRUE ,FALSE}
    }
};

Instruction MatrixTexts[MATRIXSCREENS]=
{
    "Clear screen",
    "Move ahead",
    "Move ahead, keep right",
    "Turn around, move ahead, keep left",
    "Turn around, move ahead",
    "Turn around, move ahead, keep right",
    "Move ahead, keep left",
    "Stop and wait for assistance",
    "Go to refuge"
};

#endif
```

5^e alerte :

```
Matrices.h:6:1: note: (near initialization for 'ArrowStyleMatrix')
SignMatrix.c: In function 'SetElement':
SignMatrix.c:6:22: warning: unused parameter 'Line' [-Wunused-parameter]
void SetElement (int Line, int Column)
```

Dans le fichier SignCommand.c, je rajoute des (void) Line ; et (void) Column ; dans les fonctions SetElement et ResetElement car Line et Column ne sont pas utilisés :

```
#include <stdio.h>
#include "Shapes.h"
#include "Matrices.h"
#include "SignMatrix.h"

void SetElement (int Line, int Column)
//Illuminate the matrix element
{
    (void) Line;
    (void) Column;
    printf ("*");
}

void ResetElement (int Line, int Column)
//Extinguish the matrix element
{
    (void) Line;
    (void) Column;
    printf (" ");
}
```

Cela me supprime toutes les alertes restantes.

Question 3

Comme tout les langages de programmation, le C a évolué au cours des année et plusieurs révisions des normes définissant ce langages existent. Il est important dans les projets industriels de bien spécifier la norme utilisée et de s'y conformer strictement afin d'éviter les ambiguïtés qui peuvent découler de l'utilisation de normes différents dans des modules distincts du programme.

Dans ce projet, nous utiliserons la **norme C89** et nous allons demander au compilateur de nous rapporter les problèmes de respect de cette norme qu'il trouve.

Pour cela, éditer le fichier «build.bat» et ajouter l'option «-std=c89» dans la ligne de compilation.

Relancer le script de compilation et observer le résultat.

Corriger ces problèmes avant de continuer.

Alerte :

```
X:\Semestre_8\MOQL\TP1>gcc -I. -c -Wall -Wextra -Wpedantic -std=c89 Main.c MatrixDrain.c SignCommand.c SignMatrix.c
In file included from MatrixDrain.c:1:0:
MatrixDrain.h:8:27: error: C++ style comments are not allowed in ISO C90
    int BatteryCapacity=1000; //Initialised to maximum capacity per hour
                             ^
MatrixDrain.h:8:27: error: (this will be reported only once per input file)
SignCommand.c: In function 'SignCommand':
SignCommand.c:64:6: error: C++ style comments are not allowed in ISO C90
    // Follow arrows
    ^
SignCommand.c:64:6: error: (this will be reported only once per input file)
In file included from SignMatrix.c:3:0:
Matrices.h:6:2: error: C++ style comments are not allowed in ISO C90
    //Straight
    ^
Matrices.h:6:2: error: (this will be reported only once per input file)
SignMatrix.c:7:1: error: C++ style comments are not allowed in ISO C90
    //Illuminate the matrix element
    ^
SignMatrix.c:7:1: error: (this will be reported only once per input file)
```

Pour les premières alertes, j'ai simplement supprimé les commentaires dans les fichiers spécifiés.

Pour la dernière alerte, j'ai retiré les espaces après les antislash dans le fichier Matrices.h.

Exercice 2 :

Question 1

Le problème trouvé est le suivant :

Fichier	Sévérité	Ligne	Résumé	Since date	Tag
▼ Mat... Mat...	erreur de style	4	The function 'M...	31/01/2024	

CWE: 561 The function 'MatrixDrain' is never used.

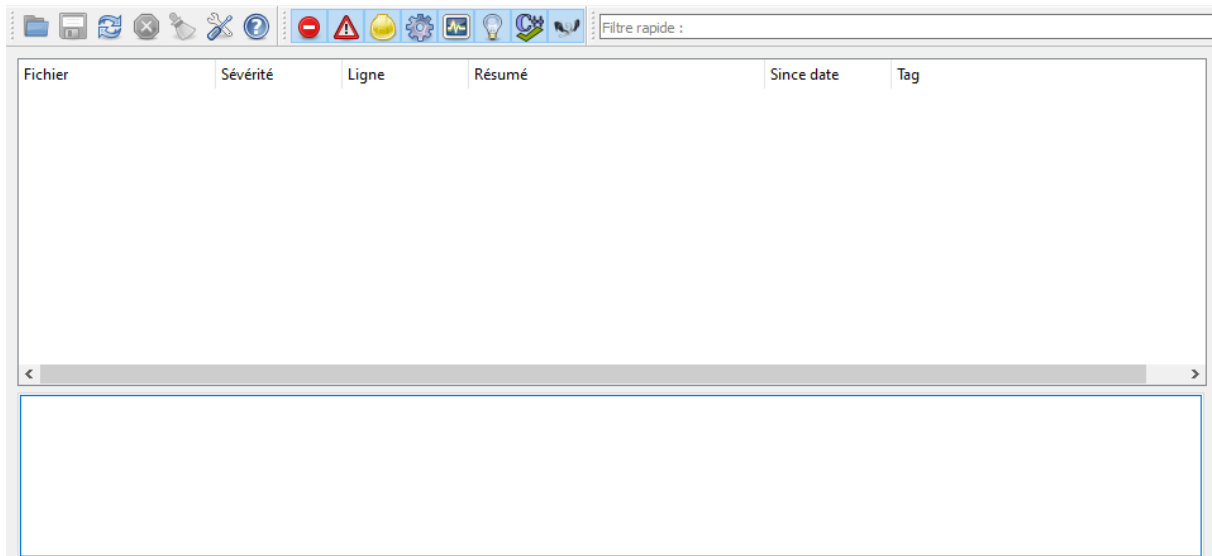
J'ai mis la fonction 'MatrixDrain' en commentaire dans le fichier MatrixDrain.c

```
#include "MatrixDrain.h"
#include "Shapes.h"

/*int MatrixDrain (enum Manufacturer Brand, int ElementCount)
{
    int Drain;

    switch (Brand)
    {
    case Lampy:
        Drain=3*ElementCount*LampyFactor(ElementCount);
        break;
    case MyWay:
        if (ElementCount<20)
            Drain=4*ElementCount;
        else
            Drain=3*ElementCount;
        break;
    case Matrixmagic:
        if (ElementCount<10)
            Drain=2*ElementCount;
        else
            if (ElementCount<20)
                Drain=3*ElementCount;
            else
                Drain=4*ElementCount;
        break;
    default:
        Drain=4*ElementCount;
    }
    BatteryCapacity=BatteryCapacity-Drain;
    return Drain;
}*/
```

En relançant l'analyse, il n'y a plus d'alerte :



Question 2

Indiquez les alertes trouvées par CPPCheck dans votre rapport en les regroupant par type d'alerte (règle MISRA non respectée).

Corriger ces alertes avant de poursuivre. Si une alerte vous semble difficile à corriger et/ou que le résultat n'améliore pas réellement la qualité du code, vous pouvez écrire une déviation à MISRA. Dans ce cas, ajouter dans le code, à l'endroit de l'alerte, un bloc de commentaire expliquant la règle MISRA à ignorer et les raisons.

Il n'y a que des erreurs de style :

Fichier	Sévérité	Ligne	Résumé	Since date	Tag
▼ Main.c					
Main.c	erreur de style	8	MISRA 17.7 (Required) The value returned...	31/01/2024	
Main.c	erreur de style	6	MISRA 8.2 (Required) Function types shall...	31/01/2024	
▼ MatrixDrain.h					
MatrixDrain.h	erreur de style	8	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
▼ SignCommand.c					
SignCommand.c	erreur de style	58	MISRA 16.4 (Required) Every switch state...	31/01/2024	
SignCommand.c	erreur de style	56	MISRA 12.1 (Advisory) The precedence of...	31/01/2024	
SignCommand.c	erreur de style	54	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	53	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	52	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	45	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	41	MISRA 10.4 (Required) Both operands of ...	31/01/2024	
SignCommand.c	erreur de style	41	MISRA 12.1 (Advisory) The precedence of...	31/01/2024	
SignCommand.c	erreur de style	41	MISRA 15.6 (Required) The body of an ite...	31/01/2024	
SignCommand.c	erreur de style	39	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	38	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	36	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	33	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	27	MISRA 8.2 (Required) Function types shall...	31/01/2024	
SignCommand.c	erreur de style	27	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	22	MISRA 8.2 (Required) Function types shall...	31/01/2024	
SignCommand.c	erreur de style	22	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	17	MISRA 8.2 (Required) Function types shall...	31/01/2024	
SignCommand.c	erreur de style	17	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	11	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	8	MISRA 17.7 (Required) The value returned...	31/01/2024	
SignCommand.c	erreur de style	6	MISRA 8.4 (Required) A compatible decla...	31/01/2024	
SignCommand.c	erreur de style	1	MISRA 21.6 (Required) The Standard Libra...	31/01/2024	
▼ SignMatrix.c					
SignMatrix.c	erreur de style	73	MISRA 15.1 (Advisory) The goto statemen...	31/01/2024	

Pour résoudre ces erreurs, dans le fichier MatrixDrain.h, j'ai modifié les lignes 7 et 8 : j'ai rajouté « extern » et j'ai supprimé la valeur numérique de BatteryCapacity.

```
#ifndef Matrixdrain_H
#define Matrixdrain_H

#include "Shapes.h"

extern int LampyFactor (int ElementCount);
extern int MatrixDrain (enum Manufacturer Brand, int ElementCount);
extern int BatteryCapacity;

#endif
```

Dans le fichier SignCommand.c, j'ai rajouté la déclaration de certaines fonctions.

Dans le fichier SignCommand.h, j'ai rajouté la ligne : void SignCommand(void)

```
#ifndef Signcommand_H
#define Signcommand_H

typedef unsigned int UINT_32;

void SignCommand(void);

#endif
```

J'ai modifié le fichier main.c :

```
#include "SignCommand.h"

int main(void)
{
    SignCommand();
    return 0;
}
```

J'ai ajouté des parenthèses pour rendre l'ordre d'évaluation explicite dans le fichier SignCommand.c :

```
if ((N <= (int)Choice) && (MaxChoice >= (int)Choice))
    UpdateDisplay(Choice);
}
```


Après ça, il me restait peu d'erreur que j'ai tenté de corriger mais en vain, j'ai donc écrit des déviations à MISRA avec le bloc de code suivant :

```
/*  
 * MISRA 9.4 (Required):  
 * An element of an object shall not be initialized more than once.  
 * This comment ensures that each element of the array is initialized only once.  
 */
```

Je me suis permis de faire ça car mes erreurs étaient affichées en tant que « erreurs de style », ce qui n'est pas critique pour faire fonctionner le code.

Question 3

Voici les erreurs que je dois corriger :

▼	SignMatrix.c		
❌	SignMatrix.c	erreur	43 Array index out of bounds, cannot determine that ThisArro... 31/01/2024
❌	SignMatrix.c	erreur	43 Array index out of bounds, cannot determine that ThisArro... 31/01/2024
❌	SignMatrix.c	erreur	127 Cannot determine that 'bForward' is initialized 31/01/2024
❌	SignMatrix.c	erreur	127 Cannot determine that 'bLeft' is initialized 31/01/2024

En premier, à la ligne 43 j'ajoute une vérification pour m'assurer que « ThisArrow » est dans la plage attendue :

```

    if (ThisArrow >= 0 && ThisArrow < SUBMATRIXTYPECOUNT)
    {
        bForward = TRUE;           \arrow != Exclaim)
        bForwardInitialized = TRUE;
        bLeft = TRUE;
        bLeftInitialized = TRUE;

        Display:

        Line=StartLine;

        do
        {
            Column=StartColumn;
            do
            {
                if (ArrowStyleMatrix[ThisArrow][Line][Column])
                    SetElement(Line,Column);
                else
                    ResetElement(Line,Column);
                Column+=ColumnIncrement;
            } while (Column!=EndColumn);

            Line+=LineIncrement;
            printf ("\n");
        } while (Line!=EndLine);
        printf ("\n");

        return;
    }

```

Pour les autres erreurs, j'ai rajouté 2 variables :

Cela me permet de savoir quand bForward et bLeft sont initialisés.

Je n'ai plus d'erreurs.

CONCLUSION

Nous avons vu dans ce TP comment utiliser le compilateur et CPPCheck pour réaliser une analyse statique d'un projet écrit en langage C afin d'en améliorer la qualité.

Cette analyse est rapide et facile à réaliser. Dans les environnements de développement professionnel, il est fréquent d'automatiser ces contrôles afin qu'ils soient exécutés à chaque compilation. De cette manière, les problèmes potentiels sont détectés dès leur écriture et avant même l'exécution du programme et le développeur peut ainsi corriger rapidement toute déviation par rapport aux normes de qualité.

Cependant, l'analyse statique ne peut détecter toutes les erreurs, et ne remplace donc pas les autres type d'analyse et les tests.