

TP 04 – Tableaux

Exercice 0 Mise en place de vos solutions

Comme vu en cours, certaines fonctionnalités intéressantes sur les tableaux ne sont disponibles qu'à partir de la norme C++17 du langage. Par défaut, Visual Studio se place en mode C++14 et ne prend donc pas en charge ces avancées. Ce premier exercice est un tutoriel pour bien configurer vos solutions Visual Studio avec la prise en charge du C++17.

Par ailleurs, tous les systèmes d'exploitation actuels pour ordinateurs personnels fonctionnent en 64 bits. Or, Visual Studio crée des solutions qui sont compatibles avec les systèmes 32 bits ET 64 bits en vous permettant de choisir le type de plateforme cible de votre programme et se place par défaut en mode 32 bits. Le tutoriel va alors également vous permettre de supprimer la compilation en 32 bits, souvent source de malentendus au niveau de la configuration des projets.

- 📁 Créez une nouvelle solution en vous basant sur le modèle *Application Console*.
- 📁 Visual vous crée alors un squelette de code source avec juste un `std::cout << "Hello World!\n" ;`.

0.1 Suppression de la compilation pour les plateformes 32 bits

En haut de la fenêtre de Visual Studio, vous devez voir deux listes déroulantes vous affichant « *Debug* » et « *x86* ». Il s'agit pour la première du mode de compilation « *Debug* » ou « *Release* » qui permet de compiler votre programme en prenant en charge le débogage (*Debug*), ou en optimisant le code pour la taille du programme et sa vitesse d'exécution (*Release*). La seconde sélectionne le type de plateforme cible : *x86* (application pour systèmes 32 bits) ou *x64* (application pour systèmes 64 bits).

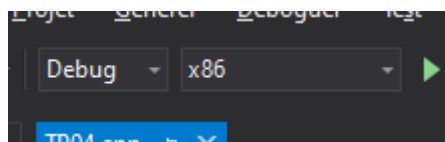


Figure 1 : Sélecteur de mode de compilation et de plateforme cible.

- 📁 Dans la liste déroulante des plateformes cibles, choisissez « *Gestionnaire de configurations...* » ou choisissez l'élément de menu *Générer / Gestionnaire de configurations...*
- 📁 Puis, dans celui-ci, faite dérouler la liste des *Plateformes de la solution active* et choisissez *<Modifier...>*.

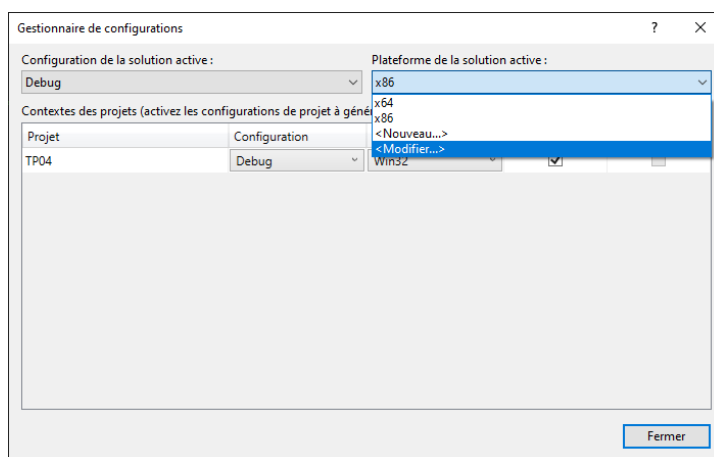


Figure 2 : Gestionnaire de configurations.

- ➡ Dans la nouvelle boîte de dialogue, supprimez la ligne « **x86** » puis fermez les deux boîtes de dialogue pour retourner à votre code source.

0.2 Activation du support du C++17

- 📁 Dans l'*explorateur de solution*, sélectionnez votre projet (attention, pas la solution mais bien le projet)

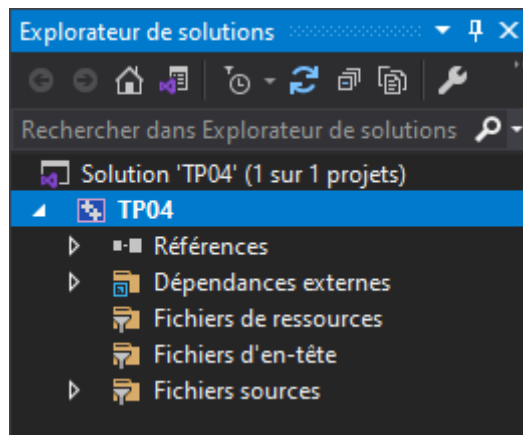


Figure 3 : Explorateur de solutions.

- 📁 Faites un clic droit sur le projet et choisissez *Propriétés...*, ou bien choisissez l'élément de menu *Projet / Propriétés...* pour afficher les *pages de propriétés* du projet.
- ➡ Dans la page de *propriétés générales*, pour la ligne *Norme du langage C++*, choisissez une norme supérieure ou égale à C++17.

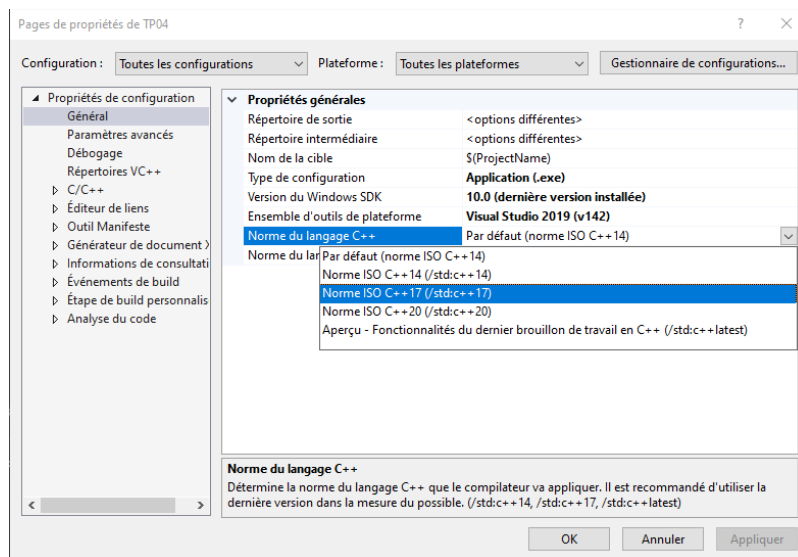


Figure 4 : Pages de propriétés du projet.

- ➡ Fermez les *pages de propriétés*.
- ➡ Votre solution est maintenant prête !

Exercice 1 Calculs mathématiques

- ✎ Écrivez un programme qui demande la saisie de 10 valeurs réelles et en calcule la moyenne et l'écart-type. Vous parcourrez le tableau en utilisant des boucles **for** de **i = 0** à **9**.
- ➡ Rappel des formules :
 - ➡ Moyenne : $\mu = \frac{1}{N} \sum_{i=1}^N x_i$
 - ➡ Écart-type : $\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$
- ✎ Ré-écrivez ce programme en parcourant le tableau en utilisant cette fois des boucles **range-for** là où cela est possible.

Exercice 2 Recherche de valeur

2.1 Saisie d'un tableau

- ✎ Écrivez un programme qui demande la saisie de 10 nombres compris entre 0 et 99.
- ➡ Vous devez valider chaque valeur et faire en sorte que si l'utilisateur a saisi une valeur non valide, la saisie de ce nombre soit réessayée.

2.2 Affichage du tableau

- ✎ Affichez le tableau sur deux lignes :
 - ➡ La première ligne affichera les indices du tableau
 - ➡ La seconde ligne affichera les valeurs du tableau
- ✎ Mettez en forme l'affichage précédent pour que les valeurs soient alignées verticalement entre elles :
 - ➡ La fonction **std::setw(3)** disponible dans la bibliothèque **<iomanip>** permet de définir la largeur en caractères de la sortie du prochain nombre, en complétant avec des espaces si nécessaire. Par exemple le code **std::cout << std::setw(4) << 12;** affichera deux espaces suivis de '12'.
 - ➡ Vous devez obtenir ce genre d'affichage :

0	1	2	3	4	5	6	7	8	9
6	75	2	14	9	9	4	25	9	42

Figure 5 : Exemple d'affichage du tableau.

- ➡ **FACULTATIF** – A ne faire que lorsque tout le sujet de TP est fini – Améliorez cet affichage pour qu'il s'affiche comme sur la figure ci-dessous :

i	0	1	2	3	4	5	6	7	8	9
tab	6	75	2	14	9	9	4	25	9	42

Figure 6 : Exemple d'affichage du tableau mis en forme.

- ➡ **SUPER FACULTATIF** – A ne faire que si vous y êtes autorisé par l'enseignant – Améliorez cet affichage pour qu'il s'affiche comme sur la figure ci-dessous :

i	0	1	2	3	4	5	6	7	8	9
tab	6	75	2	14	9	9	4	25	9	42

Figure 7 : Exemple d’affichage du tableau avec mise en forme avancée.

2.3 Recherche d’une valeur

- ☞ Ajouter une demande de saisie d’une valeur à rechercher dans le tableau
- ☞ Selon la valeur saisie, affichez les positions où cette valeur est rencontrée dans le tableau. Par exemple, avec le tableau de la figure 5, vous devez obtenir les affichages suivants selon les cas :

```
Entrez la valeur à rechercher : 9
La valeur 9 a été trouvée en position 4.
La valeur 9 a été trouvée en position 5.
La valeur 9 a été trouvée en position 8.
```

Figure 8 : Affichage du résultat de la recherche de la valeur 9 dans le tableau de la figure 5.

```
Entrez la valeur à rechercher : 99
La valeur 99 n'a pas été trouvée dans le tableau.
```

Figure 9 : Affichage du résultat de la recherche de la valeur 99 dans le tableau de la figure 5.

- ☞ Faites en sorte maintenant que lorsqu’une valeur est trouvée plusieurs fois dans le tableau, la liste d’indices soit affichée comme sur la figure suivante :
- ☞ Vous devrez probablement stocker les positions trouvées dans un tableau dynamique.

```
Entrez la valeur à rechercher : 9
La valeur 9 a été trouvée en positions 4, 5 et 8.
```

Figure 10 : Affichage amélioré du résultat de la recherche de la valeur 9 dans le tableau de la figure 5.

2.4 Détermination du minimum et du maximum

- ☞ Affichez enfin les valeurs minimales et maximales du tableau.

```
Le minimum du tableau est 2 et son maximum est 75.
```

Figure 11 : Affichage du maximum et du minimum du tableau de la figure 5.

2.5 Création d’un menu d’application

- ☞ Créez le menu d’application comme sur la figure suivante.
- ☞ Ce menu doit se réafficher à la fin de chaque action, sauf évidemment pour l’action ‘Quitter’.

```

Menu de l'application
0 : Quitter
1 : Saisir le tableau
2 : Afficher le tableau
3 : Rechercher une valeur
4 : Afficher le minimum et le maximum
Choisissez votre action :


```

Figure 12 : Menu de l'application.

 Gardez précieusement le code de cet exercice, il sera utile dans l'exercice 4.


Exercice 3 Calculs entre éléments

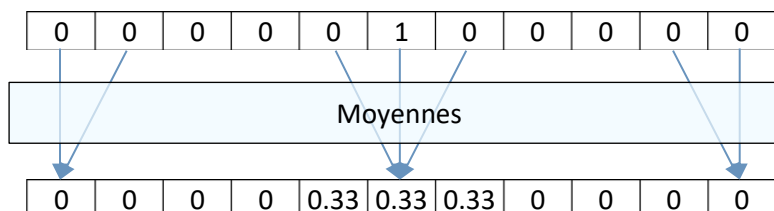
3.1 Lissage d'une valeur


-  Écrivez un programme qui initialise un tableau de N nombres réels avec tous ces éléments à 0 sauf celui du milieu égal à 1.


 Pour les exemples, nous choisirons $N = 11$. Ainsi, le tableau initialisé est :

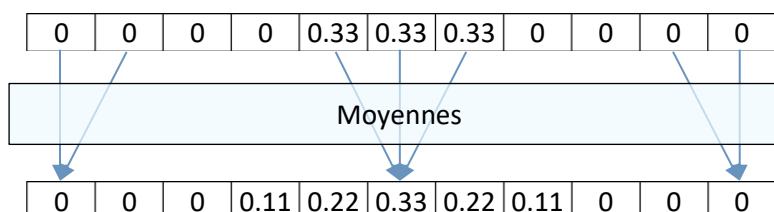
0	0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---


-  Remplacez chaque élément par la moyenne entre lui-même et ses deux voisins. Pour les éléments des bords, la moyenne ne se fera qu'avec l'élément disponible à droite ou à gauche. Ainsi, après ce traitement le tableau résultat se constitue de la façon suivante :




 Vérifiez que vous obtenez bien le résultat attendu en l'affichant dans la console.

-  Répétez l'action précédente et vérifiez que vous avez bien le résultat suivant dans notre exemple :



-  Une fois ces deux précédentes étapes réussies, introduisez-les dans une boucle de sorte que ce traitement se réalise ($4 * N$) fois.

-  Pour le tableau obtenu au bout des $4 * N$ passes du traitement, déterminez son maximum.

 Si vous êtes malin, le maximum se détermine très facilement...

3.2 Affichage du profil lissé

Nous allons considérer les valeurs numériques contenues dans les cases du tableau comme des hauteurs. Pour chaque case, nous afficherons une pile de 'o' dont la hauteur sera proportionnelle à la valeur de la case. La hauteur maximale sera de $\frac{N}{4}$ 'o'. Vous pouvez donc calculer pour chaque case sa hauteur en nombre de 'o'.

- 📖 Selon le principe qui vient d'être énoncé, réaliser l'affichage du profil du tableau obtenu lors de l'exercice 3.1.
- ➡ Pour l'exemple où $N = 11$, vous devriez obtenir la figure suivante :

```

      o
    ooooooooo
  
```

Figure 13 : Profil obtenu pour $N = 11$.

- ➡ Il est vrai que cela n'a pas l'air très impressionnant... Essayez de passer N à 200 :

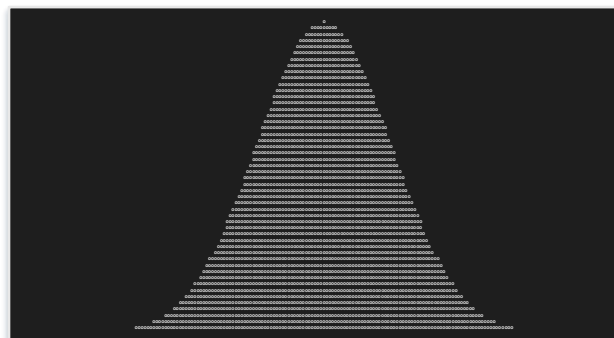


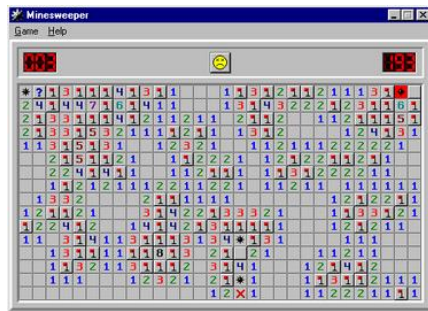
Figure 14 : Profil obtenu pour $N = 200$.

Exercice 4 Ajout / Suppression de valeurs

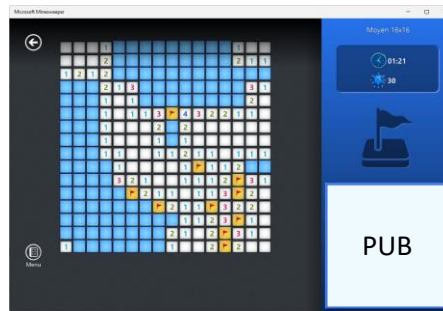
- 📖 Reprenez l'exercice 2 avec un tableau de taille dynamique. La taille du tableau sera renseignée par l'utilisateur lors de l'étape de saisie.
- ➡ Si vous avez codé correctement, vous n'avez que très peu de modifications à faire.
- 📖 Ajoutez l'élément de menu « **5 : Ajouter une valeur** » pour lequel vous demanderez la position où insérer l'élément ainsi que sa valeur et modifierez le tableau en conséquence.
- ➡ Validez correctement les entrées, c'est-à-dire assurez-vous que la position est bien comprise en 0 et la taille du tableau et que la valeur saisie est bien comprise entre 0 et 99.
- 📖 Ajoutez l'élément de menu « **6 : Supprimer une valeur par sa position** » pour lequel vous demanderez la position de l'élément à supprimer et modifierez le tableau en conséquence.
- ➡ Validez correctement les entrées, c'est-à-dire assurez-vous que la position est bien comprise en 0 et la taille du tableau moins 1.
- 📖 Ajoutez l'élément de menu « **7 : Supprimer une valeur** » pour lequel vous demanderez la valeur des éléments à supprimer et modifierez le tableau en supprimant tous les éléments qui ont cette valeur.
- ➡ Validez correctement les entrées, c'est-à-dire assurez-vous que la valeur saisie est bien comprise entre 0 et 99.

Exercice 5 Le démineur

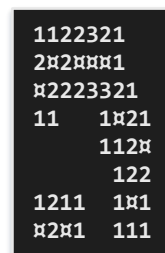
Vous allez implémenter les prémices d'un jeu du démineur.



Windows 95/98



Windows 11



Notre TP

Figure 15 : Évolution de démineur de Windows 95 à aujourd'hui

5.1 Taille de plateau fixe

Notre programme sera muni d'un tableau d'entiers à deux dimensions. Chaque case du tableau représentera une case du plateau de jeu du démineur. Les cases ayant la valeur 9 seront les mines, les autres cases auront une valeur comprise entre 0 et 8, correspondant au nombre de mines qui touchent la case.

📁 Créez un tableau 2D de taille 9x9 dont toutes les valeurs sont nulles (pas de mines).

Nous allons maintenant placer aléatoirement les mines sur le plateau. Pour tirer un nombre aléatoire, on peut utiliser le code suivant :

```
#include <ctime>
#include <cstdlib>

int main()
{
    //Initialisation du moteur de génération de nombre aléatoire :
    //(A ne faire qu'une seule fois au début du programme)
    std::srand(std::time(0));
    //Pour choisir un nombre entier aléatoire compris entre min et max (inclus)
    //on peut faire :
    int min = 2;
    int max = 8;
    int aleat = static_cast<int>(static_cast<double>(max - min + 1)
        * std::rand() / (RAND_MAX + 1)) + min;
}
```

Code 1 : Extrait de code permettant la sélection d'un nombre aléatoire

- 📁 En tirant parti de l'extrait de code 1, placer aléatoirement 10 mines sur le plateau du jeu.
 - ➡ Vérifiez votre travail en affichant le plateau :
 - ➡ Affichez des espaces à la place des '0'.
 - ➡ Affichez des 'M' à la place des '9' (les mines).
- 📁 Créez l'algorithme permettant pour chaque case de connaître combien de mines la touchent.

- 🖥️ En utilisant le même algorithme d’affichage que précédemment, affichez le plateau pour vérifier votre implémentation.

5.2 Taille de plateau variable

- 📖 Modifiez le code précédent afin que la taille du plateau soit renseignée par l’utilisateur, ainsi que le nombre de mines.
- 🖥️ Vérifiez votre code par l’affichage du plateau

🖥️ A partir de ce point, libre à vous de finir le jeu et de soigner son affichage.

📖 Pour coder les cases découvertes de celles qui ne le sont pas, vous pouvez simplement ajouter la valeur 10 à toutes les cases. Quand la valeur est comprise entre 0 et 9, la case est découverte, quand la valeur est comprise entre 10 et 19, la case est cachée.

- 🖥️ Vous pouvez également reprendre l’exercice 2.2 pour améliorer votre affichage du tableau.