SAÉ S1.02 – Le compte est bon

Nom de la Compétence visée : Compétence 1 - Comparaison d'approches algorithmiques.

Description des objectifs : En partant d'un besoin exprimé par un client, il faut réaliser une implémentation, comparer plusieurs approches pour la résolution d'un problème et effectuer des mesures de performance simples. Cette SAÉ permet une première réflexion autour des stratégies algorithmiques pour résoudre un même problème.

Apprentissages critiques : Analyser un problème avec méthode, Comparer des algorithmes pour des problèmes classiques.

Types de livrable ou de production :

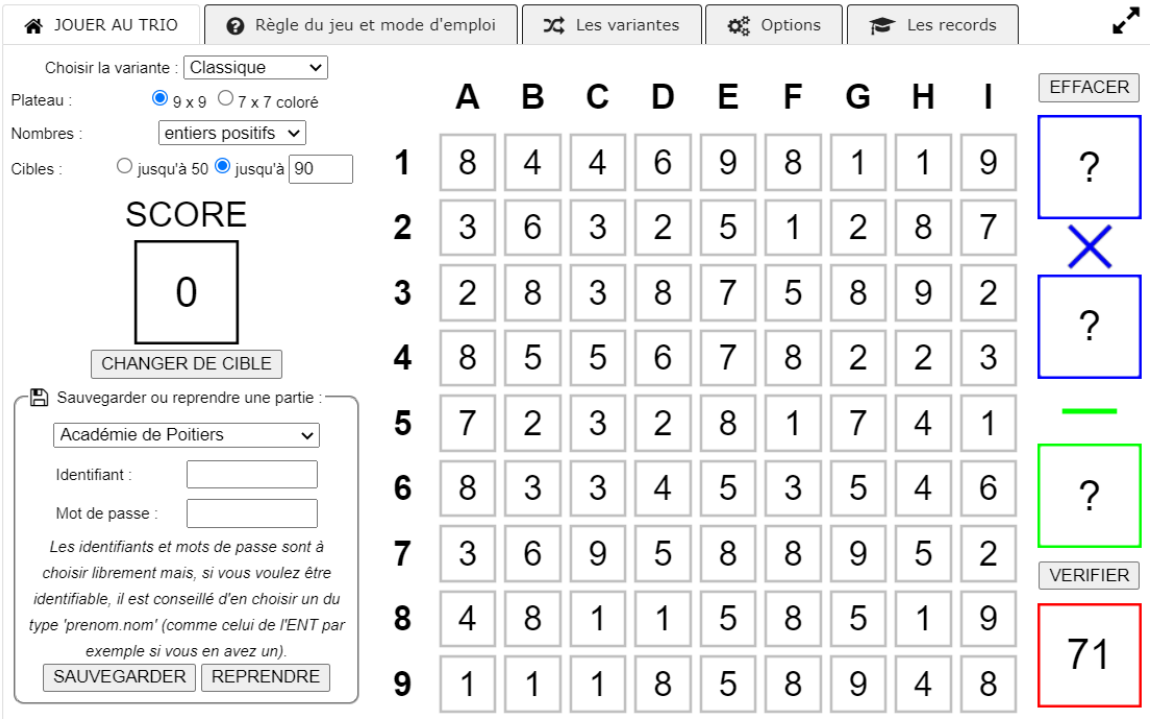
– Code de l'application ;

– Présentation du problème et de la comparaison des différentes approches.

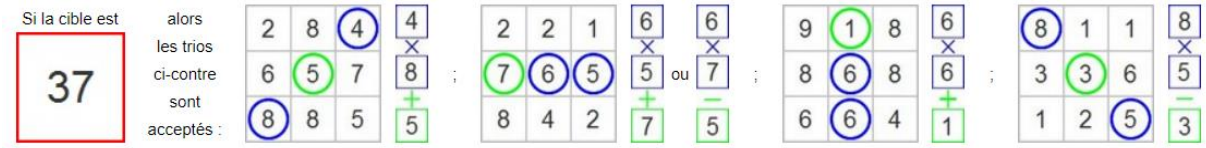
Vous êtes en stage dans une entreprise proposant des jeux éducatifs. Cette entreprise veut proposer un jeu de calcul mental du type « Le compte est bon ! », appelé TRIO.

Une version en ligne de ce jeu est disponible ici :

http://ww2.ac-poitiers.fr/math/spip.php?article1024



Principe du jeu



Comment jouer au trio classique ?

L'objectif de ce jeu est de trouver 3 nombres : les deux premiers nombres, qui doivent être multipliés entre eux, puis le troisième nombre qui sera additionné ou soustrait au résultat de cette multiplication. Ces trois nombres sont forcément ***contigus et alignés***, ***horizontalement***, ***verticalement*** ou en ***oblique***.

Description du projet

Le but de ce projet est d'implémenter un algorithme permettant de trouver la ou les solutions éventuelles pour une grille de jeu et un nombre cible donnés.

Vous disposez pour cela d'un programme de base qui lit un fichier de configuration (fichier texte) (Figure 1) contenant la taille et les valeurs d'une grille et qui extrait cette grille dans un tableau à

2 dimensions (Figure 2). Vous n'avez pas à faire vous-même cette extraction, nous vous fournissons ce code. Vous allez donc commencer à travailler à partir d'un tableau d'entiers à 2 dimensions. Les tableaux contiennent forcément le même nombre de lignes que de colonnes. La taille maximale des grilles est de 10.

Initialisation de la grille

L'initialisation des nombres sur la grille se fera à partir d'un fichier texte qui va contenir :

- La taille de la grille
- Tous les nombres contenus dans cette grille.

Le code qui extrait et qui affiche la grille de nombre est déjà écrit, vous ne devez pas le faire. Plusieurs exemples de grilles vous sont fournis, de la taille 3x3 à la taille 10x10.

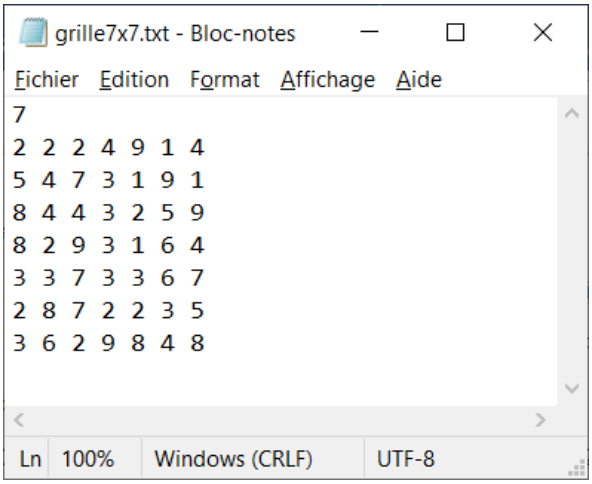


Figure 1: Fichier texte contenant les nombres contenus dans la grille

Grille extraite à partir du fichier texte grille7x7.txt :

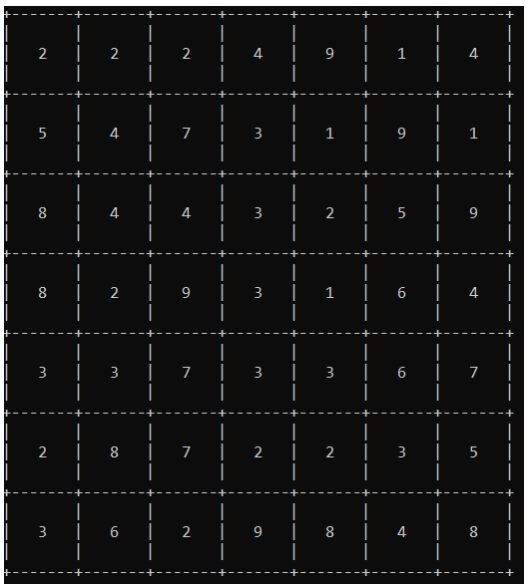


Figure 2: Grille extraite et affichée dans la console

Il y a deux modes possibles pour ce jeu :

- Mode 1 (*Figure 3*) : votre programme demande une valeur cible à l'utilisateur, et il doit trouver trois nombres A, B et C de la grille (alignés et contigus) tels que : A * B + C = cible ou A * B – C = cible. Si plusieurs calculs permettent de trouver la cible demandée, le premier que votre programme trouve est affiché.

- Mode 2 (*Figure 4*) : votre programme demande une valeur à l'utilisateur, détermine tous les calculs possibles à partir d'une grille donnée. Les calculs doivent être classés par ordre croissant. Il affiche ensuite tous les calculs qui permettent de trouver la cible demandée. Exemple, pour une cible de 5 :

- 1 × 2 + 3

- 2 × 1 + 3

- 2 × 4 – 3

- 3 × 3 – 4

- 4 × 2 – 3

- 3 x 3 - 4.

Attention, il faut bien afficher A x B +/- C et B x A +/- C, certaines configurations qui semblent identiques peuvent ainsi apparaître 2 fois.

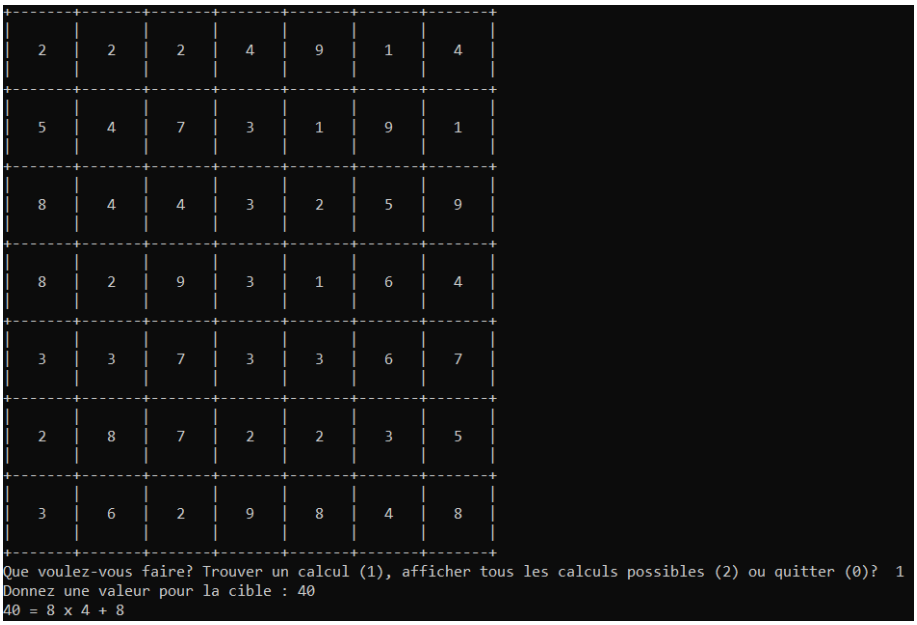


Figure 3: Exemple d'exécution du programme pour le mode 1

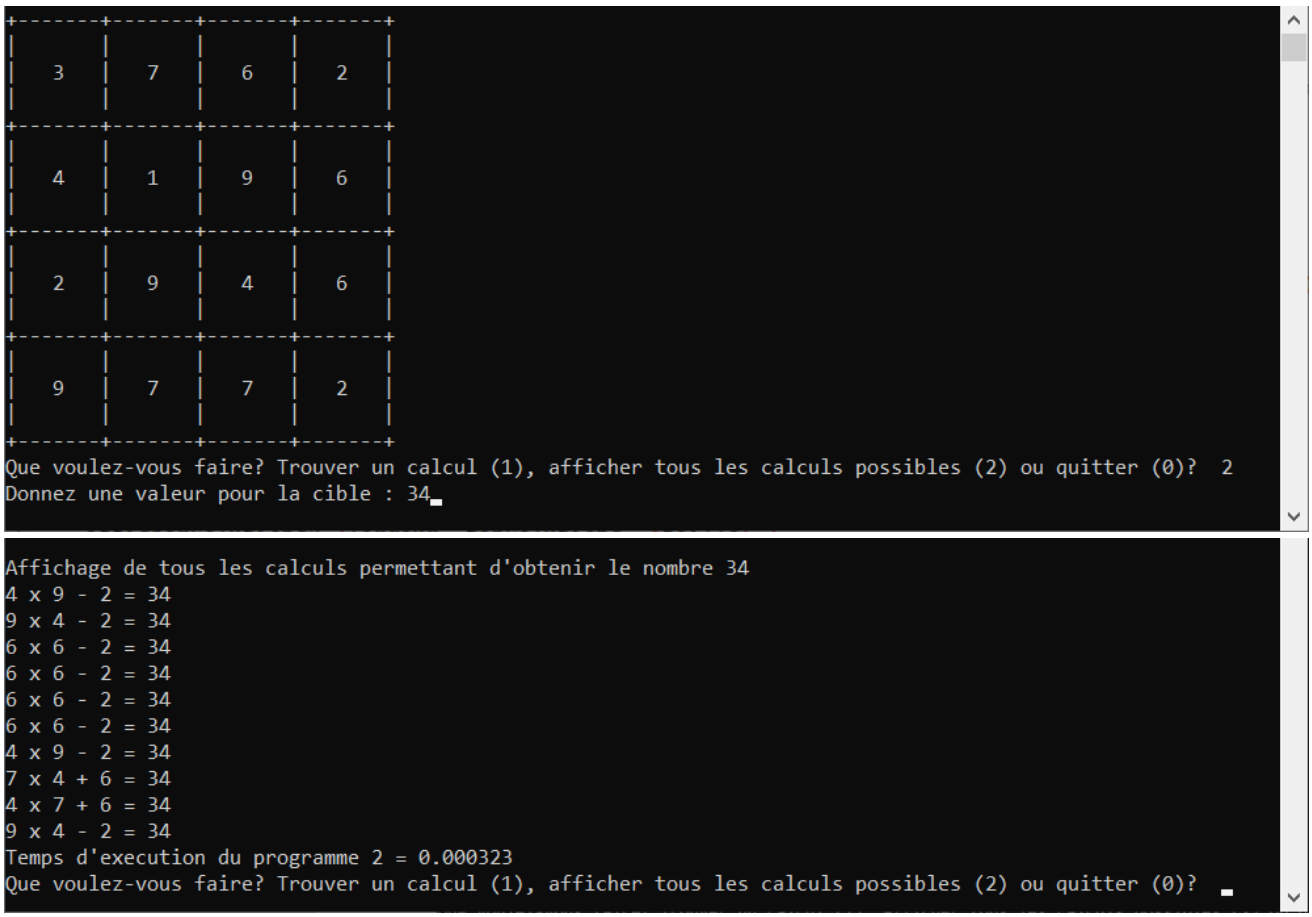


Figure 4: Exemple d'exécution du programme pour le mode 2

```
FILE    *fGrille ;
int     largeurGrille ;
int     grille [10][10] ;
int     cible ;
clock_t tempsDebut1, tempsFin1 ;
float   tempsCpu1 ;
clock_t tempsDebut2, tempsFin2 ;
float   tempsCpu2 ;
int     choixMenu = 0 ;
int         taille = 0 ;
int         tailleMax = (int)pow (6, largeurGrille+1) ;
solution    tabSolutions [tailleMax] ;
```

Le programme principal gère les variables suivantes :

o fGrille : variables qui permettent d'accéder aux fichiers de configuration initiale de la partie,

o grille et largeurGrille : variables qui permettent de gérer l'affichage de la grille dans la console. largeurGrille est une valeur entière comprise entre 3 et 10

o cible : valeur qui va être utilisée pour le calcul. Cette valeur doit être comprise entre 0 et 90,

o tempsDebut1, tempsFin1, tempsCPU1 : variables qui permettent de calculer le temps d'exécution du programme 1,

o tempsDebut2, tempsFin2, tempsCPU2 : variables qui permettent de calculer le temps d'exécution du programme 2,

o choixMenu : variable qui récupère le choix de l'utilisateur pour la version du jeu à exécuter,

o taille : variable réelle du tableau qui va contenir l'ensemble des calculs possibles avec une grille donnée,

o tailleMax : taille maximale du tableau qui va contenir tous les calculs possibles dans une grille donnée. Cette valeur est calculée pour être suffisamment grande pour pouvoir contenir tous les calculs possibles,

o tabSolutions : tableau qui permet de stocker tous les calculs possibles dans une grille donnée. Pour cela, on utilise une structure de données particulière :

```
typedef struct
{
    int     a ;
    int     b ;
    int     c ;
    int     cible ;
    char    op ;
} solution ;
```

o a, b, c : les 3 chiffres qui sont impliqués dans l'opération,

o cible : le nombre que l'on cherche à retrouver dans la grille

o op : opération associée au chiffre c. Cette opération peut être soir l'addition soit la soustraction.

Contraintes

Vous devrez travailler par groupes de 4. Vous devrez travailler à partir :

- Du fichier `main.c` qui contient le programme principal qui va extraire les grilles des fichiers textes et qui va ensuite appeler les procédures `version1` et `version2` qui correspondent aux deux modes présentés dans la partie précédente.

- De grilles pré-remplies, de la taille 3×3 jusqu'à la taille 10×10.

- De la structure de données suivante (qui sera utilisée pour la version 2) :

```
typedef struct
{
    int     a ;
    int     b ;
    int     c ;
    int     cible ;
    char    op ;        /// vaut + ou -
} solution ;
```

Vous ne devez pas modifier le fichier `main.c` ni les différents fichiers texte contenant les grilles, mais vous pouvez créer vos propres fichiers texte pour gérer vos propres grilles.

Vous devrez fournir :

- Un fichier ***entetesVersion1.h*** qui contient tous les prototypes des fonctions et procédures nécessaires pour la version 1. Ce fichier vous est fourni avec seulement le prototype de la procédure version1, vous devrez rajouter les prototypes de toutes les fonctions et procédures que vous serez amenés à créer.

- Un fichier ***version1.c*** qui contient toutes les fonctions et procédures nécessaires pour la version 1,

- Un fichier ***entetesVersion2.h*** qui contient tous les prototypes des fonctions et procédures nécessaires pour la version 2. Ce fichier vous est fourni avec seulement le prototype de la procédure version2, vous devrez rajouter les prototypes de toutes les fonctions et procédures que vous serez amenés à créer.

- Un fichier ***version2.c*** qui contient toutes les fonctions et procédures nécessaires pour la version 2.