Rapport – Jour 1 (TP6 Advent of Code 2025) – Langage C

1) Objectif

Le coffre possède un cadran circulaire numéroté de 0 à 99, avec une flèche initialement positionnée sur 50.

Le fichier d'entrée contient une série de rotations, une par ligne :

R d : tourner vers les nombres plus grands

L d : tourner vers les nombres plus petits

où d est la distance (un entier).

Deux questions sont posées :

Combien de fois la flèche termine exactement sur 0 après une rotation ?

Combien de fois la flèche passe sur 0 pendant toutes les rotations, même si elle ne s'arrête pas dessus ?

2) Gestion du cadran circulaire

Le cadran est circulaire :

Aller à droite depuis 99 revient à 0

Aller à gauche depuis 0 revient à 99

En C, cela se gère avec l'arithmétique modulaire (% 100), ce qui garantit que la position reste toujours dans [0, 99].

3) Question 1 : position finale sur 0

a) Principe

Pour chaque rotation :

Droite (R) : pos = (pos + distance) % 100

Gauche (L) : pos = (pos - distance + 100) % 100

L'ajout de +100 évite les valeurs négatives, car le modulo sur des nombres négatifs en C peut produire un résultat inattendu.

b) Exemples

Droite : R110 → (50 + 110) % 100 = 10

Gauche : L30 → (50 - 30 + 100) % 100 = 120 % 100 = 20

c) Comptage

Après chaque rotation complète, si pos == 0, on incrémente un compteur.

d) Lecture du fichier

fopen() pour ouvrir 01.input

fgets() pour lire ligne par ligne

fclose() pour fermer le fichier

e) Résultat

Pour le fichier 01.input, la réponse est : 1040.

4) Question 2 : passages sur 0 pendant la rotation

a) Différence avec Q1

Ici, on compte chaque passage sur 0 pendant la rotation, pas seulement la position finale.

Exemple : si la flèche est à 50 et effectue R1000, elle fait 10 tours complets et passe donc 10 fois sur 0.

b) Simulation pas à pas

Pour chaque rotation :

Si droite (R) → pas = +1

Si gauche (L) → pas = -1

Répéter distance fois :

avancer d’un cran

renormaliser avec % 100

si pos == 0, incrémenter count_pass_zero

c) Avantages et limites

✓ Simple à comprendre et à vérifier

✓ Permet de compter exactement les passages sur 0

⚠ Limite : si les distances sont très grandes, la boucle peut devenir coûteuse. Une approche mathématique pourrait optimiser ce calcul.

5) Débogage et vérifications

Test avec l’exemple fourni

Utilisation de printf pour afficher la direction, la distance et la position avant/après rotation

Vérification des grands déplacements et de l’importance du modulo

6) Conclusion

La principale difficulté réside dans la circularité du cadran. L'utilisation de % 100 assure des positions toujours valides entre 0 et 99.

Q1 : mise à jour de la position après chaque rotation et comptage des positions finales sur 0

Q2 : simulation pas à pas pour compter tous les passages sur 0, même sans arrêt