RAPPORT PROJET INFORMATIQUE :

🐾 ChenYI-TECH :

Réalisé par : Atcchaya, Espéra, Adam

CY TECH, MI1

Maelle Van Kets

SOMMAIRE

1.Introduction :

Dans le cadre de notre formation en première année de prépa intégrée (préING1), nous avons été amenés à réaliser un projet de programmation en langage C. Ce projet s'inscrit dans une logique pédagogique visant à renforcer notre compréhension des bases du développement logiciel, à travers la mise en œuvre d'une application concrète répondant à un besoin réel ou simulé.

Nous avons ainsi choisi de développer ChenYl-TECH, une application de gestion d'un chenil. Ce projet a pour objectif de simuler le fonctionnement informatique d'un refuge pour animaux, dans lequel on peut enregistrer les animaux, gérer les adoptions, effectuer des inventaires, ou encore calculer les rations alimentaires quotidiennes. Le programme fonctionne via une interface en ligne de commande et manipule les données à partir d'un fichier texte.

L'intérêt de ce projet repose sur plusieurs aspects essentiels de l'informatique :

- La manipulation de structures de données,
- La lecture et l'écriture dans des fichiers,
- La modularisation du code,
- La gestion des erreurs utilisateur,
- Et la mise en place d'un processus de compilation propre (Makefile).

Par ailleurs, ce projet a été réalisé en groupe, ce qui nous a permis de développer également des compétences en collaboration, en communication, ainsi qu'en organisation de projet.

Ce support a pour but de documenter l'ensemble du processus de développement de *ChenYl-TECH* : de la conception initiale à l'implémentation finale, en passant par la répartition des tâches, les choix techniques, les difficultés rencontrées et les solutions apportées

2. Présentation du projet :

2.1 Objectif général

L'objectif principal du projet *ChenYl-TECH* est de concevoir une application en langage C permettant de gérer efficacement les données d'un chenil. Le logiciel doit permettre à un utilisateur de :

- Consulter l'inventaire des animaux présents dans le refuge,
- Ajouter un nouvel animal dans le système,
- Adopter (retirer) un animal,
- Générer un rapport sur les rations alimentaires journalières,
- Trier les animaux selon leur âge (quartiles),
- Et afficher des informations précises sur chaque animal.

2.2 Contexte simulé

Dans le cadre de cette simulation, nous considérons que toutes les données sont stockées dans un fichier texte nommé animaux.txt. Ce fichier contient les informations essentielles pour chaque animal : identifiant (ID), espèce, nom, âge, poids et commentaire descriptif.

Le projet ne repose pas sur une base de données ou une interface graphique, mais uniquement sur une interface texte, ce qui met l'accent sur la logique de programmation et le traitement des données en C.

2.3Contraintes techniques

- **Langage** : le projet est entièrement codé en C **Compilation** : l'application est compilée à l'aide d'un Makefile, avec gestion des dépendances
- **Fichiers** : toutes les données sont stockées dans un seul fichier texte.
- **Interface** : ligne de commande et beaucoup de fichier.

3. Répartition des tâches :

Le projet *ChenYl-TECH* a été réalisé en groupe de trois personnes : **Adam**, **Espéra** et **Atcchaya**. Même si nous avons tous un niveau relativement similaire en informatique et que nous avions des difficultés, nous avons su unir nos efforts pour mener à bien le développement du projet.

Plutôt que de nous répartir les tâches de manière strictement individuelle, nous avons choisi une approche collaborative : **chaque fonction du programme a été discutée, comprise, puis codée ensemble**. Cette méthode de travail nous a permis de mieux comprendre le fonctionnement global du projet et de progresser collectivement.

Face aux difficultés rencontrées, nous n'avons jamais abandonné. Nous avons pris le temps de chercher des solutions ensemble, et lorsque cela était nécessaire, nous avons également sollicité de l'aide extérieure (enseignants, ressources en ligne, documentation) pour mieux comprendre certains concepts ou corriger nos erreurs. Ce travail d'équipe nous a permis d'éclaircir de nombreuses zones d'ombre, notamment en ce qui concerne la gestion des fichiers, les pointeurs, et l'organisation du projet en modules.

En plus de la répartition des tâches liées à la programmation, **Atcchaya s'est occupé de l'intégration du code dans l'environnement de développement** (VS Code), ainsi que de la gestion du code source sur **GitHub**. Cela a permis de centraliser le travail de chacun, de gérer les versions et de garantir une **mise à jour régulière** et **synchronisée** des fichiers du projet.

La collaboration a été un véritable pilier de notre réussite. Elle a permis de garantir la **cohérence du code**, le **respect des bonnes pratiques du langage C**, et une **intégration fluide** des différentes fonctionnalités dans un programme final fonctionnel.

4.Conception du Programme :

La conception d’un programme informatique constitue une étape essentielle dans le développement d’un projet. Elle permet de poser les bases logiques et techniques nécessaires au bon fonctionnement de l’application. Dans le cadre de notre projet *ChenYl-TECH*, la conception a été guidée par des objectifs de clarté, d'efficacité et de simplicité. Nous avons cherché à créer un programme capable de gérer les informations d’un refuge, tout en respectant les contraintes imposées par le langage C et la structure de données que nous avons choisie.

Avant de commencer le codage, nous avons réfléchi aux différentes fonctionnalités nécessaires à une gestion complète : ajouter un animal, adopter un animal, afficher un inventaire, calculer les besoins alimentaires, etc. Une fois ces fonctionnalités identifiées, nous les avons organisées sous forme de modules (fichiers .c séparés), chacun responsable d'une partie précise du programme. Cette organisation modulaire permet une meilleure lisibilité du code et facilite la collaboration en groupe.

Nous avons défini une structure Animal contenant les champs nécessaires : identifiant, espèce, nom, âge, poids et un commentaire. Toutes les données sont stockées de façon persistante dans des fichiers texte. Le fichier animaux.txt enregistre tous les animaux actuellement présents dans le refuge, tandis que le fichier adoptes.txt est utilisé pour stocker les animaux ayant été adoptés. Ce système de fichiers textes nous a permis de simuler une petite base de données sans recourir à des outils externes.

Pour structurer correctement le code et permettre l'utilisation des fonctions entre les différents fichiers .c, nous avons créé plusieurs fichiers d’en-tête (.h). Ces fichiers contiennent les déclarations de fonctions, les constantes, les structures et les inclusions nécessaires. Par exemple, nous avons un fichier utilitaires.h pour les fonctions de vérification (comme estEntier ou estNomValide), et d’autres comme ajouterAnimal.h, affichage.h ou adopterAnimal.h pour les fonctions spécifiques à chaque module. Cela garantit une meilleure séparation des responsabilités et permet au compilateur de lier les fichiers plus efficacement.

Enfin, pour faciliter la compilation de notre programme, nous avons mis en place un fichier Makefile. Celui-ci automatise la compilation en indiquant les commandes nécessaires pour compiler chaque fichier source et générer l’exécutable final. Il inclut aussi une commande make clean qui permet de supprimer les fichiers objets (.o) et

l’exécutable, assurant un environnement propre avant chaque recompilation. L'utilisation du Makefile a été très bénéfique, notamment pour gagner du temps et éviter les erreurs de compilation manuelle.

La conception a également impliqué la création de nombreuses fonctions utilitaires, rassemblées dans des fichiers dédiés, afin de centraliser les outils réutilisables et éviter les doublons dans le code. Un soin particulier a été apporté à la gestion des erreurs, notamment lors des lectures/écritures dans les fichiers, ou lors de la saisie des utilisateurs. Des contrôles ont été intégrés pour s’assurer que les données entrées sont valides, évitant ainsi des plantages ou des incohérences.

Tout au long du développement, nous avons effectué de nombreux tests manuels afin de valider chaque fonctionnalité séparément avant de les intégrer dans l’ensemble du programme. Ce travail itératif nous a permis d’assurer une certaine stabilité et cohérence du code.

En conclusion, cette phase de conception a été centrale dans notre projet. Elle nous a permis de mieux comprendre l’importance de la rigueur en programmation, du travail en équipe, et de la planification des tâches. Même si nous avons rencontré des difficultés, cette étape nous a beaucoup appris et nous a permis de créer un programme fonctionnel, utile et bien structuré.

5.Fonctionnalités principales :

Le programme *ChenYl-TECH* propose un ensemble de fonctionnalités essentielles pour gérer efficacement les animaux d'un chenil. Celles-ci ont été pensées pour répondre à tous les besoins de base d'un refuge : ajout, suivi, suppression et consultation d'animaux. Chaque fonctionnalité est organisée en module autonome, ce qui favorise une maintenance simple et une évolution progressive du projet.

A. Ajout d'un animal

Cette fonctionnalité permet à l'utilisateur d'ajouter un nouvel animal dans le fichier animaux.txt. L'utilisateur saisit différentes informations : espèce, nom, âge, poids et un commentaire. Des contrôles sont appliqués pour vérifier la validité des données (ex. : le nom ne doit pas dépasser 19 caractères et être alphanumérique, l'âge doit être compris entre 0 et 100, etc.). Un identifiant unique est automatiquement attribué à chaque nouvel animal.

B. Adoption d'un animal

L'adoption d'un animal permet de retirer celui-ci du fichier animaux.txt pour l'inscrire dans le fichier adoptes.txt, qui conserve un historique. Cette opération est déclenchée via l'ID de l'animal. Le programme vérifie si l'ID existe, puis modifie les fichiers en conséquence. Cela assure un bon suivi des animaux ayant quitté le refuge.

C. Affichage de l'inventaire avec analyse par tranche d'âge

Le programme affiche tous les animaux présents dans le chenil à partir du fichier animaux.txt. En plus de cette liste, une analyse statistique est réalisée pour répartir les animaux selon **quatre tranches d'âge**, appelées **quartiles**. Cela permet de mieux comprendre la structure d'âge de la population animale du refuge.
Le programme affiche :

- Le **nombre total** d'animaux présents.
- Le **nombre d'animaux** dans chaque quartile (de l'âge le plus jeune au plus âgé).
  Cette vue globale aide à prendre des décisions (ex. : besoin en soins, adoptions ciblées, etc.).

D. Gestion du quotidien – Calcul des rations alimentaires

- Le programme inclut une fonctionnalité appelée **DAY_FOOD** qui calcule la **quantité totale de croquettes nécessaires pour** nourrir tous les animaux en une journée, selon les règles suivantes :
- **Hamster** : 20 g par jour
- Lapin : 2,5 kg (soit 2500 g) par jour
- **Chat ou chien** :
    - Si l'animal a **moins de 2 ans**, il reçoit **500 g** de croquettes
    - Sinon, il reçoit **10 % de son poids** en croquettes (ex. : un chien de 30 kg reçoit 3 kg, soit 3000 g)

E. Recherche d'un animal

Une fonction de recherche permet de retrouver un animal selon son nom ou son espèce. Elle est pratique pour répondre aux questions des visiteurs ou pour vérifier rapidement la présence d'un animal.

F. Contrôle des saisies

L'ensemble du programme est conçu pour éviter les erreurs de saisie. Toutes les entrées de l'utilisateur sont analysées et contrôlées avant d'être traitées, ce qui améliore la robustesse du programme. Des messages d'erreur clairs sont affichés en cas de problème.

G. Organisation modulaire du code

Les fonctionnalités sont réparties dans plusieurs fichiers .c (comme ajouterAnimal.c, adopterAnimal.c, affichage.c, etc.) avec leurs fichiers d'en-tête .h correspondants. Cela permet une organisation claire et modulaire du projet. La compilation est gérée automatiquement grâce à un fichier Makefile, qui facilite le développement et la maintenance du code.

H. Fichiers de sauvegarde

Toutes les données sont enregistrées dans des fichiers texte persistants :

- animaux.txt contient la liste des animaux présents.
- adoptes.txt contient ceux qui ont été adoptés.
  Cela garantit la conservation des données entre chaque exécution du programme.

6- Difficultés rencontrées

Lors de la réalisation de **ChenYl-TECH**, nous avons rencontré plusieurs difficultés techniques et organisationnelles. L'une des principales a été la **compréhension du cahier des charges**, notamment pour la gestion des fichiers et l'affichage des quartiles. Nous avons dû reformuler certaines exigences pour les rendre plus claires.

La gestion des fichiers animaux.txt et adopt.txt a également posé problème. Nous avons rencontré des difficultés pour **enregistrer et manipuler correctement les données**, éviter d'écraser les informations et gérer les lectures/écritures dans ces fichiers.

En termes techniques, nous avons souvent fait face à des **erreurs de compilation** liées à des fonctions non déclarées ou des fichiers en-tête mal inclus. Ces problèmes ont nécessité plusieurs corrections de notre part, ainsi que des recherches pour mieux comprendre la structure d'un projet en C.

Lorsque nous avons rencontré des blocages, nous avons **sollicité de l'aide extérieure** pour nous aider à débuger et résoudre certains problèmes complexes. Cette assistance nous a permis de mieux comprendre les erreurs et de progresser plus rapidement.

La coordination dans notre groupe a parfois été un défi, mais elle nous a permis de mieux collaborer et d'apprendre ensemble. Atcchaya a pris en charge l'organisation du code sur Visual Studio Code et l'utilisation de GitHub pour gérer les versions.

Malgré ces difficultés, nous avons réussi à avancer et à renforcer nos compétences en programmation et en travail d'équipe.

7. Tests et résultats :

Pour assurer la qualité et la fiabilité du programme **ChenYl-TECH**, nous avons effectué plusieurs **tests unitaires** et **tests d'intégration**. Ces tests ont permis de vérifier que chaque fonctionnalité du programme fonctionnait correctement et répondait aux exigences du cahier des charges.

Les tests unitaires ont porté principalement sur les fonctions de gestion des animaux, telles que l'ajout d'un animal, le comptage des animaux dans le fichier, et la gestion des différentes tranches d'âge. Nous avons également testé la gestion des fichiers animaux.txt et adopt.txt, en vérifiant que les données étaient correctement sauvegardées, lues et mises à jour.

Concernant la fonctionnalité de calcul des croquettes pour les animaux, nous avons testé différentes configurations d'animaux (hamsters, chats, chiens, lapin) pour nous assurer que les quantités de croquettes étaient calculées correctement en fonction de l'âge et du poids.

Nous avons également testé **chaque fonction, cas par cas**, avec des saisies **valides et invalides**, pour vérifier si le programme gérait correctement les entrées incorrectes sans provoquer de crashs. Cela a permis d'identifier et de corriger les éventuelles erreurs dans la gestion des entrées utilisateurs et des calculs.

Pour les tests d'intégration, nous avons veillé à ce que toutes les fonctionnalités du programme interagissent correctement entre elles, sans conflits ou erreurs. Nous avons également testé le programme sur différents ordinateurs pour garantir sa portabilité et son bon fonctionnement.

8. Conclusion :

En conclusion, le projet **ChenYl-TECH** a été une expérience enrichissante et formatrice. Nous avons réussi à concevoir un programme répondant aux besoins du cahier des charges, tout en développant des compétences solides en programmation C et en gestion de projet.

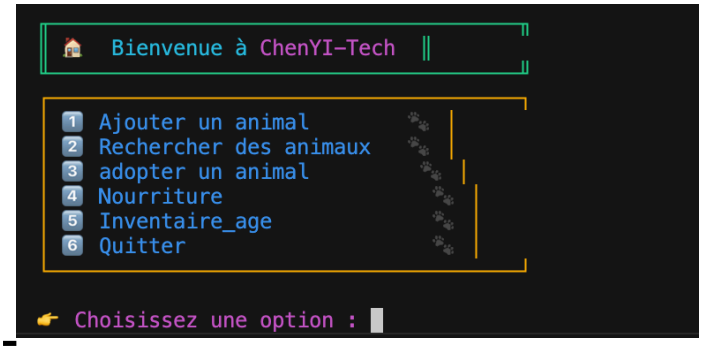
Les principales difficultés rencontrées ont été liées à la gestion des fichiers et à l'intégration des différentes fonctionnalités. Toutefois, grâce à l'entraide au sein du groupe et à l'aide extérieure que nous avons sollicitée, nous avons su surmonter ces obstacles et mener à bien le projet.

Nous avons appris à travailler en équipe, à organiser et à structurer le code de manière optimale, et à tester rigoureusement notre programme pour nous assurer qu'il fonctionne correctement. Ce projet nous a également permis de mieux comprendre l'importance de la gestion des fichiers et de la collaboration dans un projet de développement logiciel.

Nous sommes satisfaits du résultat final et pensons que **ChenYl-TECH** constitue une base solide pour une gestion optimisée des refuges animaliers.

Annexes :

MENU :

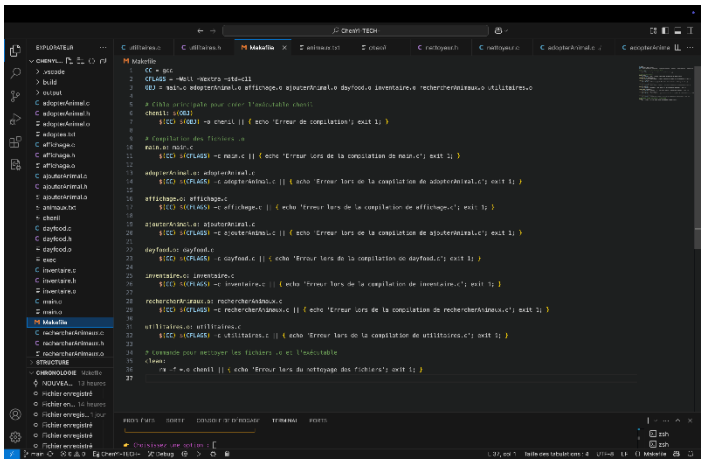


ANIMAUX.TXT

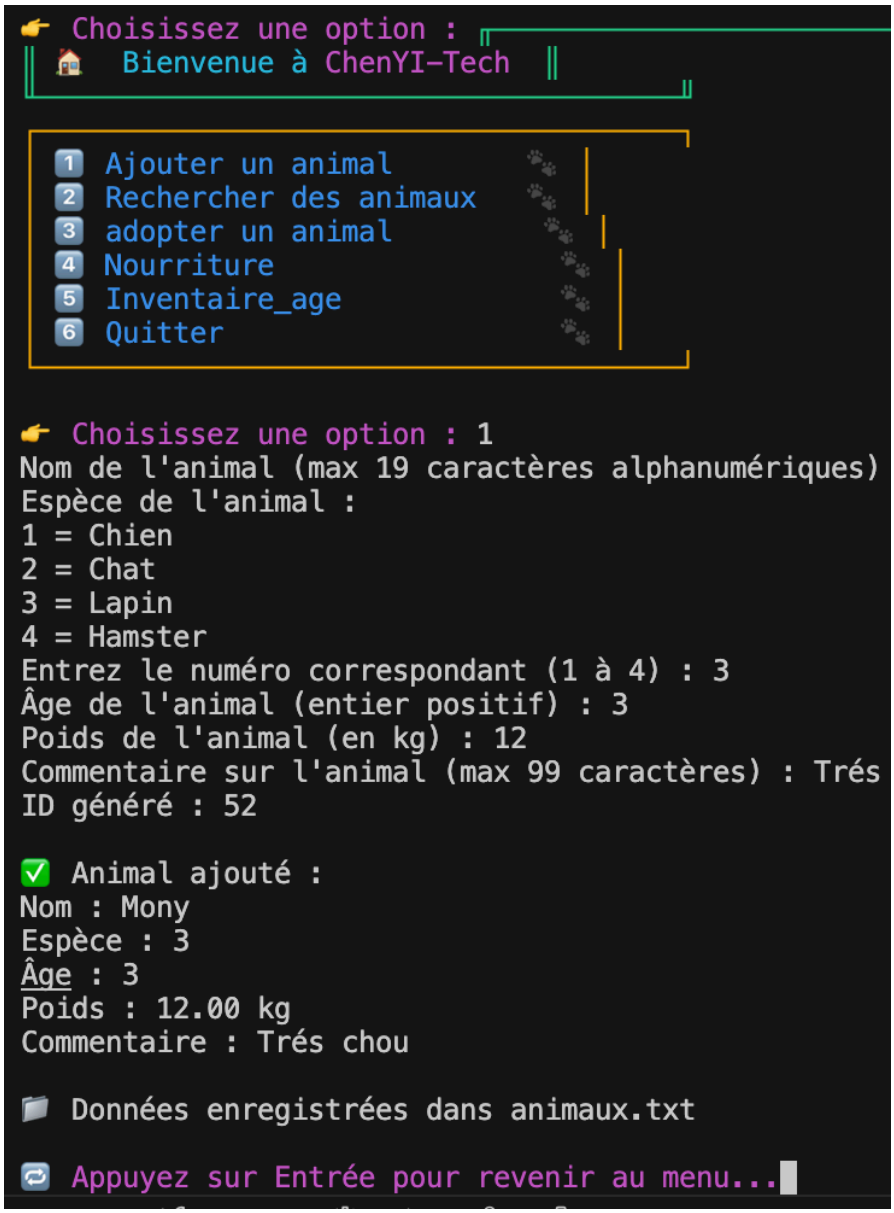
animaux.txt

```
2 2 Milou 3 8.0 Petit chien énergique
3 3 Val 1 43.0 Très gentil et affectueux
4 1 Max 12 30.0 Très protecteur et intelligent
5 4 Leo 0 6.5 Très curieux et aventureux
6 2 Mia 2 4.2 Chienne calme et douce
7 3 Nala 5 7.0 Très agité
8 3 Simba 5 27.8 Lionceau en pleine forme
9 2 Nala 11 9.9 Très loyale et calme
10 4 Coco 1 3.1 Petit mais dynamique
11 1 Rocky 4 32.3 Très robuste et protecteur
12 2 Luna 14 7.7 Très affectueuse et calme
13 3 Felix 0 5.6 Chat joueur et mignon
14 1 Bob 10 29.5 Super compagnon de promenade
15 4 Mimi 8 3.0 Très gentille et discrète
16 3 Tom 11 12.3 Très curieux et affectueux
17 2 Jerry 1 6.2 Petit mais très vif
18 1 Spike 2 33.3 Très fidèle et protecteur
19 4 Olaf 13 4.4 Sympathique mais timide
20 3 Zara 3 15.0 Chat calme et affectueux
21 2 Kiara 0 4.0 Chienne timide et réservée
22 1 Duke 7 40.0 Très énergique et protecteur
23 3 Hugo 9 18.5 Chat calme mais curieux
24 4 Lilo 1 2.2 Très joyeuse et dynamique
25 2 Kim 13 5.0 calme
26 4 Sony 10 34.0 trés gentille
27 3 Lily 12 25.0 Chat calme et affectueux
28 4 Bibi 2 1.9 Très mignonne et énergique
29 1 Roxy 3 28.8 Très intelligente et gentille
30 2 Maya 4 6.6 Très calme et douce
31 3 Oscar 10 11.0 Chat énergique et joueur
32 1 Toby 1 21.5 Très curieux et joyeux
33 4 Nina 3 3.3 Petite et calme
34 2 Sami 6 7.2 Chien énergique et joueur
35 3 Hugo 2 23.0 Très curieux et affectueux
36 1 Kim 15 36.0 Très fidèle et protecteur
37 4 Pika 0 2.0 Très énergique et curieuse
38 2 Lola 8 9.5 Très douce et câline
39 3 Nino 7 16.0 Très joueur et affectueux
40 1 Paco 1 27.0 Très amical et intelligent
41 4 Zoe 1 2.3 Très calme et adorable
42 2 Nino 3 8.2 Très amical et actif
```

Makefile:



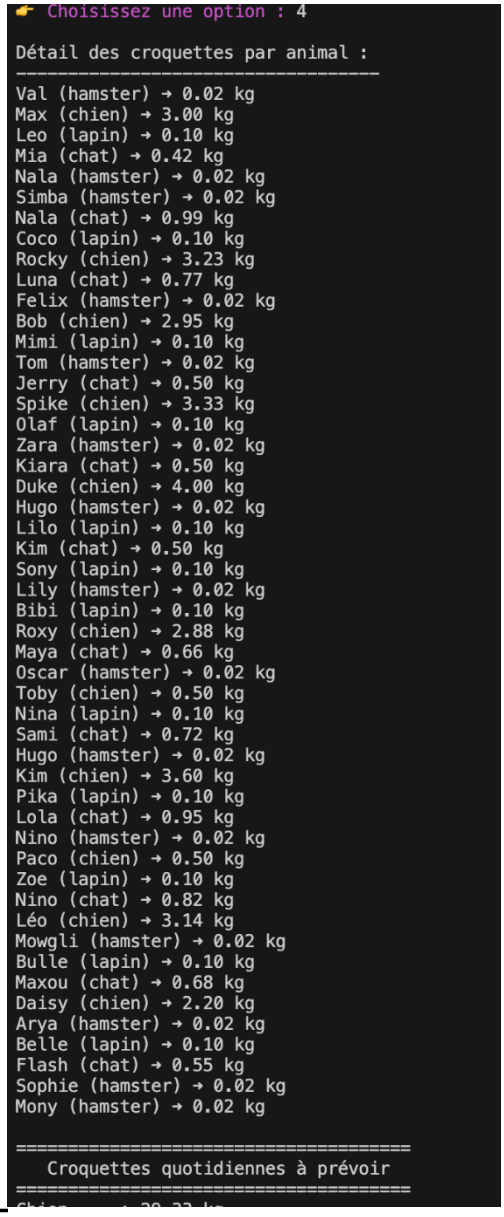
AJOUTER ANIMAUX :

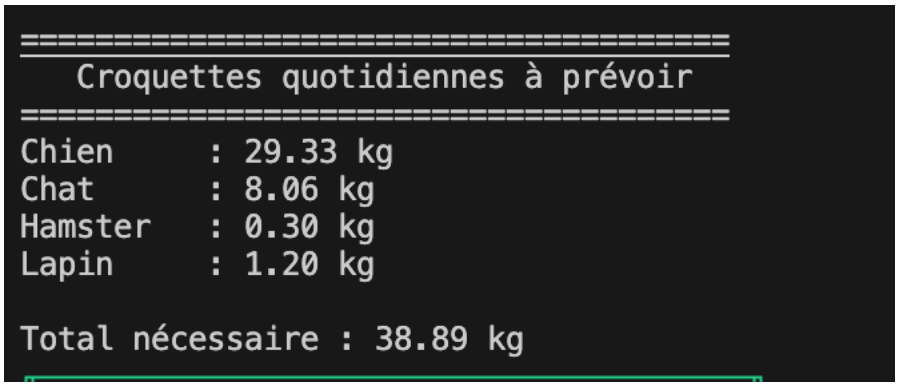


Rechercher animaux :

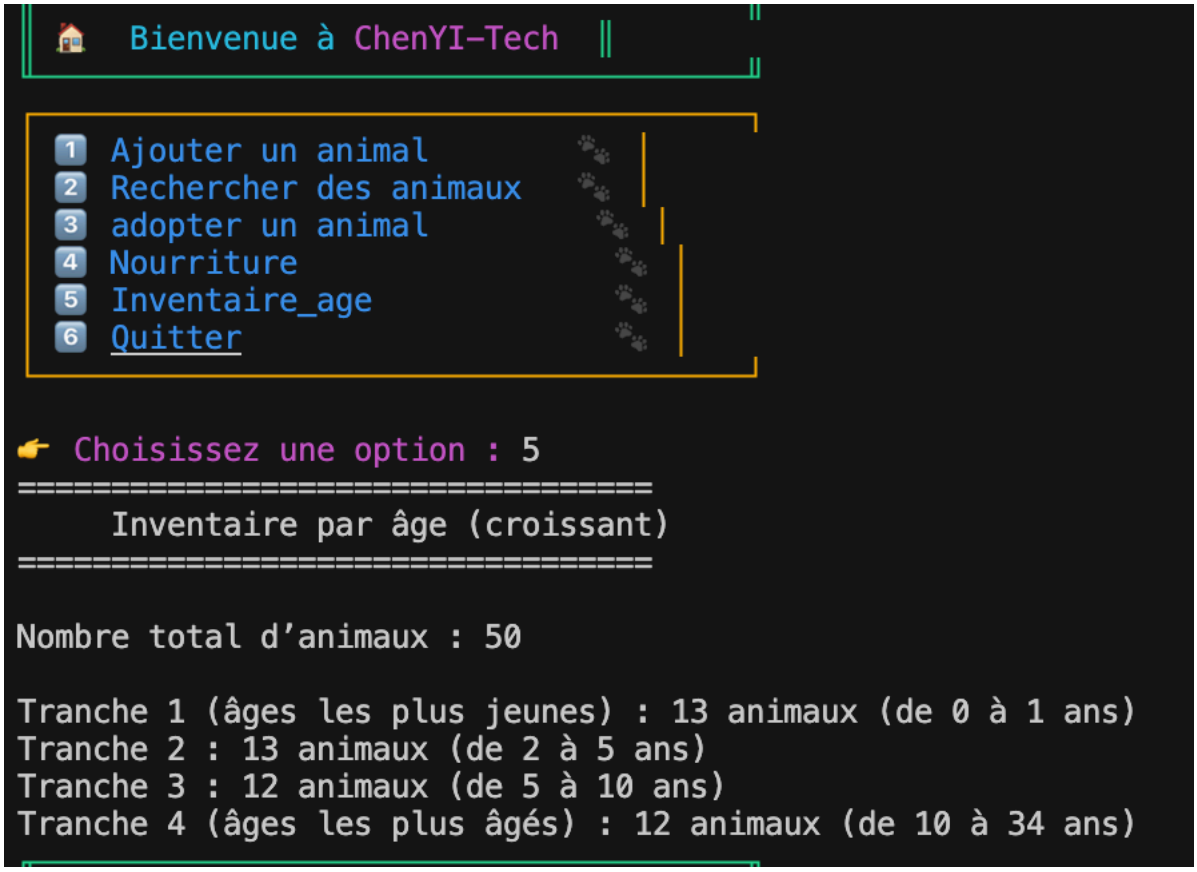
ADOPTER ANIMAUX :

NOURRITURES :





INVENTAIRES_AGE :



QUITTER: