Mini Projet

valant évaluation dans le cadre de :

Diplôme : Master Mathématiques Appliquées, 1ème année
Année universitaire : 2023-2024
Module d'enseignement : MA0844
Responsable : Francois LEFEVRE
Comptant pour : 3

Finalisé le : 29 avril 2024
Page(s) : 7
Références(s) : 0
Figure(s) – Table(s) : 0 – 0
Théorème(s) : 0

Programmation hétérogènes (C et python)

Uriel Macaire uriel-macaire.ollomo-mba@univ-reims.fr
OLLOMO MBA

Mots-clés : programmation hétérogenes, C, pyhton.

Résumé : Un langage de programmation hétérogène est conçu pour faciliter le développement de programmes qui s'exécutent sur des systèmes hétérogènes, c'est-à-dire des systèmes composés de multiples types de processeurs ou d'unités de traitement. Dans ce compte rendu, nous allons réaliser un logiciel qui articule des codes écrits en langages de programmation hétérogènes (C et Python). Les parties écrites en C et en Python devront se compléter et communiquer à partir d'un fichier ASCII. Nous assurerons également que le fonctionnement de ce logiciel soit indépendant du système d'exploitation.

Table des matières

1. Introduction

À partir de l'exercice 3 du TP3, qui manipule un fichier texte (MesNotes.txt) et établit ses statistiques (écart-type, moyenne, etc.) dans un tableau numérique, nous allons créer un logiciel qui utilise un langage de programmation hétérogène (C et Python). Pour ce faire, nous allons tout d'abord énoncer et définir les objectifs de notre logiciel ; effectuer une étude algorithmique des tâches afin de préciser celles qui pourront être réalisées en C et celles qui seront réalisées en Python ; implémenter en langage C la partie adéquate (choisie) et en Python la partie complémentaire ; et enfin réaliser la documentation de notre logiciel.

2. TP3: Statistiques dans un tableau numérique

2.1. Ennoncé

Le logiciel que nos allons creé est conçu pour analyser un ensemble de données sous forme de notes afin de fournir des insights quantitatifs à travers des statistiques descriptives et des visualisations. Les données proviennent initialement d'un fichier texte qui nécessite un traitement préliminaire pour assurer la validité et la fiabilité des informations utilisées pour les analyses subséquentes.

2.2. Objectif du logiciel

L'objectif principal de ce logiciel est de nettoyer et de filtrer les données brutes afin de les préparer pour des analyses plus approfondies. Le logiciel vise également à fournir des mesures statistiques de base, telles que la moyenne et l'écart type, pour aider à comprendre la distribution des notes. Il est conçu pour gérer efficacement les erreurs, évitant ainsi les interruptions et fournissant des messages d'erreur utiles aux utilisateurs. En outre, il garantit une fonctionnalité uniforme sans nécessiter de modifications sur divers systèmes d'exploitation, y compris Windows, MacOS et Linux.

3. Etude algorithmique et taches de chaque partie (C et Python)

les tâches réalisées en C sont principalement axées sur le traitement initial des données et la préparation de l'environnement pour l'exécution ultérieure des scripts Python.Plusieurs tâches spécifiques ont été réalisées en Python, tirant parti de ses capacités pour la manipulation de données, les calculs statistiques et la visualisation.

3.1. Tâches à effectuer en C

1. Lecture et Validation des Données

- *Ouverture et Lecture de Fichiers* :
  - Fichier Source : Le programme commence par ouvrir et lire le fichier `MesNotes.txt`, contenant les notes des étudiants.
  - Validation : Il vérifie que les données lues sont des nombres réels, conformes aux attentes.
- *Filtrage des Données* :
  - Critères de Filtrage : Le programme filtre les données pour s'assurer que seules les notes valides sont retenues, notamment celles qui ne dépassent pas 512 entrées.
  - Fichier de Sortie : Les notes validées sont écrites dans un nouveau fichier `FilteredNotes.txt`, prêtes pour l'analyse Python.

2. Gestion des Erreurs

— *Gestion des Erreurs de Fichier*:

— Ouverture et Lecture : Des vérifications sont effectuées pour assurer l'ouverture et la lecture correctes des fichiers. Les erreurs sont gérées avec `perror` pour un retour détaillé.
— Contrôle des Limites du Tableau
— Capacité Maximale : Le programme surveille le nombre d'entrées pour éviter de dépasser la limite de 512 notes. En cas de dépassement, le processus est interrompu proprement.

3. Communication avec le Module Python

— *Exécution du Script Python*:

— Lancement Automatique : Après la préparation des données, le programme C utilise `system()` pour exécuter le script Python qui procédera à l'analyse statistique et à la visualisation des données.

4. Optimisation et Performance

— *Optimisation de la Manipulation de Fichiers* :

— Efficacité de Lecture/Écriture : Le programme est optimisé pour un traitement rapide et efficace des fichiers.

— *Utilisation Efficace de la Mémoire* :

— Gestion de la Mémoire : Le code C gère soigneusement la mémoire dynamique, évitant les fuites lors des opérations de lecture et d'écriture.

3.2. Tâches à effectuer en Python

1. Chargement des Données

— Lecture du fichier : Utilisation de Python pour lire le fichier `FilteredNotes.txt`, qui contient les notes validées par le module C. Ce fichier est chargé dans un tableau NumPy pour un traitement rapide et efficace.

2. Calcul des Statistiques

— Statistiques descriptives : Emploi de NumPy pour calculer la moyenne (`np.mean()`) et l'écart type (`np.std()`) des données. Ces fonctions permettent de réaliser des calculs statistiques de manière optimisée sur de grandes quantités de données.

3. Visualisation des Données

— *Création de graphiques* :

— Utilisation de Matplotlib et Pylab pour générer des visualisations graphiques des données analysées, y compris des histogrammes et d'autres types de graphiques pour une meilleure compréhension de la distribution des notes.
— Utilisation de `Axes3D` pour des visualisations tridimensionnelles, offrant une perspective plus détaillée et visuellement engageante des analyses statistiques.

4. Gestion des Erreurs

— Traitement des exceptions: Python gère les erreurs potentielles avec des blocs `try-except`, assurant une gestion efficace des erreurs durant la lecture des fichiers, les calculs et la visualisation. Cette approche prévient les arrêts brusques du programme et fournit des retours appropriés à l'utilisateur.

5. Intégration avec le Module C

— Appel système : Utilisation de `os.system()` pour exécuter le script Python depuis le programme C, permettant de lancer les analyses statistiques et les visualisations après que les données aient été préparées par le module C.

4. Implémentation

4.1. Langage C

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    FILE *fp_read, *fp_write;
    double note;
    int count = 0;

    fp_read = fopen("MesNotes.txt", "r");
    if (!fp_read) {
        perror("Erreur lors de l'ouverture du fichier MesNotes.txt");
        return EXIT_FAILURE;
    }

    fp_write = fopen("FilteredNotes.txt", "w");
    if (!fp_write) {
        perror("Erreur lors de l'ouverture du fichier FilteredNotes.txt");
        fclose(fp_read);
        return EXIT_FAILURE;
    }

    while (fscanf(fp_read, "%lf\n", &note) != EOF) {
        if (count < 512) {
            fprintf(fp_write, "%.2f\n", note);
            count++;
        } else {
            fprintf(stderr, "Erreur: Le nombre de notes dépasse la limite autorisée de 512.\n"
            fclose(fp_read);
            fclose(fp_write);
            return EXIT_FAILURE;
        }
    }

    fclose(fp_read);
    fclose(fp_write);
    system("python3 process_notes.py");  // Lancer le script Python
    return EXIT_SUCCESS;
}
```

4.2. Langage Python

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def load_data(filepath):
    data = np.loadtxt(filepath)
    return data

def calculate_statistics(data):
    mean = np.mean(data)
    std_dev = np.std(data)
    return mean, std_dev

def plot_data(data):
    # Création de la figure
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Calcul de l'histogramme des données
    hist, bins = np.histogram(data, bins=10)

    # Calcul des positions de départ pour les barres (x, y, z)
    xpos = bins[:-1]  # Positions de départ sur l'axe x
    ypos = np.zeros_like(xpos)  # Toutes les barres commencent à y = 0
    zpos = np.zeros_like(xpos)  # Toutes les barres commencent à z = 0

    # Dimensions des barres
    dx = np.ones_like(hist) * (bins[1] - bins[0])  # Largeur des barres
    dy = np.ones_like(hist) * 0.5  # Profondeur des barres arbitraire
    dz = hist  # Hauteur des barres selon les comptages d'histogramme

    # Ajout des barres au graphique
    ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color='b', zsort='average')

    # Afficher le graphique
    plt.show()

def main():
    try:
        data = load_data("FilteredNotes.txt")
        mean, std_dev = calculate_statistics(data)
        print(f"Moyenne: {mean:.2f}, Écart type: {std_dev:.2f}")
        plot_data(data)
    except Exception as e:
        print(f"Erreur: {str(e)}")

if __name__ == "__main__":
    main()
```

5. Documentation du logiciel

Ce logiciel permet de traiter un ensemble de notes stockées dans un fichier texte, de calculer des statistiques telles que la moyenne et l'écart type, et de visualiser les résultats. Il est conçu pour fonctionner de manière indépendante sur Windows, macOS et Linux.

1. Fonctionnalités

— Lecture de données depuis un fichier.
— Filtrage et validation des données.
— Calcul de statistiques basiques.
— Visualisation des données.

2. Installation

2.1. Prérequis

Compiler C (GCC pour Linux/macOS, MSVC pour Windows). Python 3.x. Bibliothèques Python : NumPy, Matplotlib.

2.2. Instructions d'installation

— Cloner ou télécharger le code source: taper dans le terminal la commande `git clone https://github.com/macaireollomo/miniprojet.git`
— Compiler le programme C dans un terminale ou dans un logiciel tel que code Blocks

3. Utilisation

3.1. Exécution du Programme

— Pour démarrer le traitement des notes, utilisez la commande suivante dans le terminal (l'invite de commande): `./notes_processor` ou executer le code C tout simplement dans un logiciel tel que code Block.

3.2. Format des Données

— Le fichier `MesNotes.txt` doit contenir des notes (une par ligne) au format numérique. Le nombre maximum de lignes (notes) est de 512.

4. Architecture du Logiciel

4.1. Modules C

— `notes_processor.c`: Lit et valide les données du fichier MesNotes.txt, écrit les données valides dans `FilteredNotes.txt`, et lance le script Python.

4.2. Modules Python

— `process_notes.py`: Calcule la moyenne et l'écart type avec les données de FilteredNotes.txt et génère un histogramme 3D des notes.

5. Gestion des Erreurs

Le logiciel gère plusieurs types d'erreurs :

```
- Erreurs d'ouverture de fichier.
- Dépassement de la limite de notes.
- Erreurs de format de données.
- Problèmes lors de l'exécution des scripts.
```

6. Bibliographie

— logiciels utilisés: spyder, code blocks et Rstudio (Rmarkdown).
— langages de programmation: python, C/C++ et Latex
— Hebergeur: Github https://github.com/macaireollomo/miniprojet.git