Aspect	Problème potentiel	Test de validation	Méthode
Analyse du fichier CSV	Fichier CSV manquant ou corrompu	Vérification de l'existence du fichier CSV et de son intégrité	Utilisez les fonctionnalités de manipulation de fichiers en C++ pour vérifier si les fichiers existent et lisez leur contenu pour assurer qu'ils ne sont pas vides.
	Format de données invalide	Vérification du format des données CSV selon les spécifications	Utilisez les fonctionnalités de lecture de fichiers en C++ pour lire les fichiers CSV ligne par ligne, puis utilisez des expressions régulières ou d'autres méthodes appropriées pour valider le format des données.
	Absence de données	Vérification si le fichier CSV contient des données	Vérifier si le fichier CSV est vide et si l'application gère cette situation correctement
	Données dupliquées	Vérification si les données CSV contiennent des doublons	Lisez les lignes de données des fichiers CSV et utilisez une table de hachage ou une autre structure de données pour suivre les lignes de données déjà rencontrées afin de vérifier les doublons.
	Taille excessive du fichier CSV	Gérer efficacement les fichiers CSV de grande taille	Créez un fichier CSV volumineux, puis testez les performances et la stabilité de l'application lors de la lecture et du traitement de ce fichier.
	Données en dehors de la plage attendue	Vérification si les valeurs des mesures sont plausibles (ex. latitude>90)	Écrivez des fonctions de test pour vérifier si les valeurs de mesure sont dans la plage attendue, par exemple, vérifiez si la latitude est comprise entre -90 et 90 degrés.
	Incohérence entre les fichiers CSV	Vérification de la cohérence entre les données de différents fichiers	Comparez les champs de données correspondants dans différents fichiers pour garantir leur cohérence.

Aspect	Problème potentiel	Test de validation	Méthode
Analyse des données	Erreurs d'analyse ou d'interprétation des données	Vérification de la précision des résultats d'analyse	Comparer les résultats d'analyse avec des données connues pour vérifier leur exactitude
	Manque de cohérence entre les résultats	Vérification de la cohérence des résultats d'analyse	Exécuter plusieurs analyses sur les mêmes données et vérifier si les résultats sont cohérents
	Calcul de la moyenne de la qualité de l'air dans une zone circulaire	Calculer la moyenne de la qualité de l'air dans une zone spécifiée pour une période de temps donnée	Créer une zone circulaire virtuelle et une période de temps, calculer la qualité de l'air et vérifier si le résultat correspond aux attentes
	Évaluation et classement des capteurs en fonction de la similarité	Évaluer et classer les autres capteurs en fonction de la similarité par rapport à un capteur sélectionné	Vérifier si le résultat correspond aux attentes
	Détection des données non fiables provenant des particuliers	Analyser les données fournies par les capteurs individuels pour déterminer leur fiabilité	Créer des données de capteurs virtuels provenant d'utilisateurs individuels, les analyser et vérifier si elles sont fiables. Vérifier si le résultat correspond aux attentes
Gestion de la mémoire	Fuites de mémoire	Utilisation d'outils de débogage pour surveiller la mémoire	Exécutez l'application et utilisez des outils de débogage de mémoire en C++, tels que Valgrind, pour vérifier les fuites de mémoire ou d'autres problèmes.
	Surallocation de mémoire	Vérification si la mémoire est allouée de manière adéquate	Écrivez des fonctions de test pour simuler l'allocation et la libération de mémoire de l'application, puis vérifiez si son comportement est conforme aux attentes.

Aspect	Problème potentiel	Test de validation	Méthode
Efficacité de l'algorithme	Temps d'exécution excessif	Mesurer le temps d'exécution de chaque algorithme	Utilisez les fonctionnalités de mesure du temps en C++ pour enregistrer les temps de début et de fin de chaque algorithme, puis calculez le temps d'exécution.
	Complexité algorithmique excessive	Vérifier si les algorithmes sont optimisés et efficaces	Évaluez l'optimisation et l'efficacité de chaque algorithme en analysant leur temps d'exécution et l'utilisation des ressources.
Gestion des erreurs	Erreurs de saisie utilisateur	Validation des entrées utilisateur pour éviter les erreurs	Écrivez des fonctions de test pour simuler différentes entrées utilisateur et vérifiez si l'application les gère correctement.
	Manque de feedback sur les erreurs	Assurer que les utilisateurs sont informés des erreurs survenues	Simulez des situations d'erreur et vérifiez si les messages d'erreur appropriés sont affichés pour informer les utilisateurs.
	Logging et journalisation des erreurs	Assurer que les erreurs sont correctement enregistrées et journalisées	Testez les fonctions de journalisation d'erreurs pour assurer qu'elles enregistrent les erreurs de manière appropriée et qu'elles peuvent être suivies par les développeurs.
	Erreurs liées aux autorisations d'accès	Vérifier si les erreurs liées aux autorisations d'accès sont correctement gérées	Simulez des erreurs d'autorisation d'accès et vérifiez si l'application les traite correctement en refusant l'accès ou en affichant des messages d'erreur appropriés.
Optimisation des performances	Utilisation excessive de la CPU	Surveiller l'utilisation du processeur pendant l'exécution	Utilisez des outils de surveillance de l'utilisation du processeur pour mesurer et analyser l'utilisation du processeur par l'application pendant son exécution.

Aspect	Problème potentiel	Test de validation	Méthode
	Accès disque excessif	Surveiller les opérations d'entrée/sortie pendant l'exécution	Mesurer la fréquence et la durée des accès disque pendant l'exécution de l'application
	Utilisation excessive de la mémoire	Surveiller l'utilisation de la mémoire pendant l'exécution	Observer l'utilisation de la mémoire pendant l'exécution et identifier les zones où une optimisation est nécessaire