Débogage de Code C++

Méthodes et Outils pour Déboguer Efficacement

L3 maths Strasbourg

14 février 2025

Plan de la présentation

Introduction

Méthodes Manuelles de Débogage

Outils Automatisés de Débogage

Conclusion

Introduction au Débogage

- Le débogage consiste à identifier, isoler et corriger les erreurs dans le code.
- Plusieurs techniques et outils existent pour faciliter cette tâche.
- Nous verrons des méthodes allant de l'analyse manuelle à l'utilisation d'outils avancés.

Exécution sur papier pas à pas

- Simuler l'exécution du code à la main pour comprendre la logique.
- Identifier les variables critiques et suivre leur évolution.
- Utile pour détecter des erreurs de logique ou des cas particuliers.
- Exemple : tracer l'exécution d'une boucle ou d'une récursion sur papier.

Relecture et Analyse de Code

- Lire attentivement le code, éventuellement à haute voix ou avec un pair.
- Identifier les formes d'erreurs récurrentes (erreurs d'indexation, boucles infinies, etc.).
- Utiliser des commentaires et des noms de variables explicites pour faciliter la relecture.

Affichage de Variables et Utilisation d'asserts

- Insérer des `std::cout` pour afficher les valeurs critiques.
- Utiliser `assert` pour vérifier que certaines conditions sont remplies.

Listing 1 – Utilisation d'assert dans le code

```
#include <cassert>
#include <iostream>

int division(int a, int b) {
    assert(b != 0); // Vérifie que b n'est pas zéro
    return a / b;
}

int main() {
    int result = division(10, 2);
    std::cout << "Résultat : " << result << std::endl;
    return 0;
}
```

Valgrind

- Valgrind est un outil puissant pour détecter les fuites de mémoire et les accès invalides.
- Il vérifie que toute la mémoire allouée est correctement libérée.
- Commande de base :

```
valgrind --leak-check=full ./votre_programme
```

Sanitizers

- Les sanitizers (ASan, UBSan, etc.) sont intégrés aux compilateurs modernes.
- Ils détectent des erreurs comme les débordements de tampon, les accès hors limites, etc.
- Exemple d'utilisation avec `g++` ou `clang++` :
  - `-fsanitize=address` pour l'AddressSanitizer.
  - `-fsanitize=undefined` pour l'UndefinedBehaviorSanitizer.

Débogueurs : gdb et dbg

- **gdb** est le débogueur GNU standard pour C/C++.
- Il permet de mettre des points d'arrêt, inspecter les variables, et exécuter le code pas à pas.
- **dbg** est une interface simplifiée pour gdb, facilitant sa prise en main.

Exemple de Session gdb

Listing 2 – Session de débogage avec gdb

```
(gdb) break main
Breakpoint 1 at 0x4006a7: file main.cpp, line 10.
(gdb) run
Starting program: ./a.out
Breakpoint 1, main () at main.cpp:10
10      int x = 5;
(gdb) print x
$1 = 5
(gdb) next
11      x = x + 2;
(gdb) print x
$2 = 7
(gdb) continue
```

Conclusion et Bonnes Pratiques

- Le débogage est un processus essentiel pour garantir la qualité et la robustesse du code.
- Utiliser une approche combinée : analyse manuelle, affichage de variables, assertions, puis outils automatisés.
- La pratique régulière et l'utilisation d'outils dédiés améliorent significativement la capacité à résoudre les bugs.

Questions et Discussion

Des questions ?