1 Debugging

Nutzen sie `GDB`, um die Codes zu erkunden und die Fehler zu beseitigen.

1.1 Syntax-Fehler

Das Programm `exercise_01.c` kann im derzeitigen Zustand nicht kompiliert werden, da einige Syntax- und Semantik-Fehler im Programm vorhanden sind. Beheben Sie die Fehler und Warnungen, sodass das Programm kompiliert und korrekt ausgeführt werden kann.

1.2 Speicher-Fehler

Das Programm `exercise_02.c` ist frei von Syntax-Fehlern und lässt sich kompilieren. Allerdings sind einige Bugs im Code enthalten, die dafür sorgen, dass die Ausgabe nicht der Spezifikation entspricht oder das Programm zur Laufzeit abstürzt. Korrigieren Sie das Programm soweit, dass der tatsächliche Output der Vorgabe im Blockkommentar entspricht. Versuchen Sie dabei, die Anpassungen möglichst minimal zu halten und den zugrunde liegenden Algorithmus nicht grundlegend zu ändern.

2 Dynamischer Struct-Speicher

Implementieren Sie einen dynamischen Struct-Speicher, in dem Koordinaten-Structs abgelegt, ausgelesen und wieder entfernt werden können. Es ist Ihnen überlassen, ob Sie für die Implementierung der Datenstruktur sich an einem Array oder einer verketteten Liste oder etwas ganz anderem orientieren.
Folgende Funktionen soll Ihre Datenstruktur unterstützen:

- Erstellen und Freigeben Ihrer neuen Struct-Speicher-Datenstruktur zur Speicherung der Koordinaten-Structs
- Hinzufügen eines Koordinaten-Structs an beliebiger Position oder am Ende
- Ausgabe der Koordinaten `x`, `y`, `z` eines einzelnen Koordinaten-Structs, aller Koordinaten-Structs in einem Bereich und aller Koordinaten-Structs in der Struct-Speicher-Datenstruktur auf die Konsole
- Löschen eines einzelnen Koordinaten-Structs, aller Koordinaten-Structs in einem Bereich und aller Koordinaten-Structs
- Ausgabe der Anzahl an Koordinaten-Structs in Ihrer Struct-Speicher-Datenstruktur

Verwenden Sie als Vorlage `exercise_03.c` und überprüfen Sie Ihre Applikation mit Valgrind.

Hinweis: Viele Funktionen können durch einen angepassten Aufruf einer ähnlichen Funktion implementiert werden; Sie können damit Zeit und Arbeit sparen.

Optional:

- Verwenden Sie `callgraph` (oder eine Alternative), um Ihren Programmablauf zu visualisieren.
- Verwenden Sie `massif` (oder eine Alternative), um die Verwendung des Heap-Speichers durch Ihr Programm zu visualisieren.

Material

Makefile und Code-Vorlagen liegen dem beiliegenden Übungsmaterial bei. Sie können alle Binaries automatisch mittels `make` oder gezielt mit `make BINARYNAME` bauen lassen.