AUFGABE 2: IMPLEMENTIERUNG EINES FILTERS

Ziel dieser Aufgabe ist es, ein besseres Verständnis für die Implementierung von Funktionalitäten für Signalverarbeitung wie Filterung und die Problematik von Fließkomma-Arithmetik zu entwickeln. Hierbei soll Wert auf einen modularen Softwareentwurf gelegt werden. Die Schnittstellen sowie deren Datentypen sollen eindeutig strukturiert werden.

Die Vorgabe befindet sich im Ordner 02_filter des Vorgaben-Repositories:

https://gitlab.cs.fau.de/ezs/vezs18—vorgabe.git

Starten Sie die Anwendung mit make run im Build-Verzeichnis, nachdem sie mittels

source ./ecosenv.sh && mkdir build && **cd** build && cmake ..

dieses erstellt haben.

1 Aufgabenstellung

1. Datentypen und Signaturen In dieser Aufgaben werden wir eine Schnittstelle (`real.h`), welche die Implementierung von reellen Zahlen abkapselt, verwenden. Machen Sie sich mit dieser vertraut und erweitern Sie `real.c` um die Implementierungsvariante welche den Typ `REAL` mittels `float` implementiert.

Implementieren Sie nun die Konvertierung von `signal_input` zu `filter_input` an geeigneter Stelle. *Welche Überprüfungen der Eingabedaten sollten vorgenommen werden?*

Antwort:

2. Aufgabe

Vergleichen Sie die Signaturen der Funktionen `convolve_data()` mit `convole_batch()`.

Was fällt hierbei auf und was könnte verbessert werden? Welche Entscheidungen bei der Definition der Eingabe kann die statische WCET-Analyse der entsprechenden Verarbeitungsfunktionen erleichtern bzw. erschweren?

Antwort:

3. Verarbeitung

Implementieren Sie die Filterinitialisierung `convolve_filter_init()` und die Faltung `convolve_filter_step()`. Benutzen Sie dazu den Datentyp `REAL`.

☞ mathworld.wolfram.com/Convolution.html

Nachdem Sie den Filter implementiert haben, verwenden Sie ihn mit den entsprechenden Argumenten in `convolve_data()` und `convolve_batch()` um `signal_input` zu falten.

Nach der Ausführung sollen nun die Ergebnisse verarbeitet werden. Hierfür soll eine Prüfsumme über die Ausgabedaten generiert werden. Implementieren Sie dazu `checksum()` und berechnen Sie die Prüfsumme für `signal_input` mehrmals.

Nachdem sie eine stabile Prüfsumme ermittelt haben, sollen Sie nun eine Fehlerbehandlung implementieren, sollte die Prüfsumme falsch sein. (Dies dient zur Vorbereitung auf die folgende Übung zu Fehlerinjektionsexperimenten). *Welches Verhalten eignet sich hier im Fehlerfall? In welchen Zustand befindet sich der Filter im Fehlerfall?*

Antwort:

4. Q-Format Erweitern Sie nun die Aufgabe so, dass für die Signalverarbeitung keine Fließkomma- sondern Festkomma-Datentypen verwendet werden. Verwenden Sie dazu die vorgegebene Festkommabibliothek (`fixpoint.[h|c]`) für arithmetische Operationen mittels Q-Format.

Wenn Sie Ihr Programm strukturiert implementiert haben, müssen Sie dafür nur Änderungen in `real.h` und `real.c` vornehmen. Benutzen Sie für das Auswählen der Implementierungsvariante mit Fließkomma- oder Festkomma-Datentypen die Präprozessor-Definition (`#define FIXEDP`).

Unterscheidet sich der Wert von der Prüfsumme aus Teilaufgabe 3, wenn die Ausgabewerte in das Fließkomma-Format zurück transformiert werden? Wenn ja, warum?

Antwort:

5. Aufgabe Identifizieren Sie die Vor- und Nachteile der nicht-funktionalen Eigenschaften des Programmes bei der Verwendung von Fließ- bzw. Festkomma-Datentypen. *Welche Vor- und Nachteile konnten sie feststellen?*

Antwort:

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 03.05.2018
- Fragen bitte an `i4ezs@lists.cs.fau.de`