

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:

```
ssh://git@git.cs.tu-dortmund.de:2222/VSS-WS20/vss-vorgaben.git
```

Hinweis: Beginnend mit diesem Übungsblatt ist für die Bearbeitung der Aufgaben eine definierte Entwicklungsumgebung erforderlich. Hierfür erhalten Sie in einer gesonderten eMail Zugangsdaten zur Infrastruktur der Arbeitsgruppe und insbesondere dem Server:

`kos.cs.tu-dortmund.de`

Der Zugang erfolgt analog¹ zu den Pool-Rechnern der IRB, am einfachsten per SSH. Weitere Informationen und Zugangsmöglichkeiten mit grafischer Benutzeroberfläche finden Sie auf den Veranstaltungsw Webseiten.

```
ssh <login>  
@kos.cs.tu-dortmund.de
```

Nach den vorbereitenden Schritten (siehe Aufgabe 1), können Sie ihre Anwendung übersetzen mittels

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

und starten Sie das System im Emulator mit `make run` im Build-Verzeichnis.

Hinweis: Dieses, ebenso wie die beiden folgenden Aufgabenblätter, setzen das Echtzeit-Betriebssystem eCos ein. Wir haben Ihnen in der Vorgabe die wichtigsten, für VSS erforderlichen Schnittstellen als Kommentare im Code erläutert. Beginnen Sie mit dem Lesen in der Datei `app.c` in der Funktion `cyg_user_start`. Mehr Informationen finden Sie bei Bedarf in den Übungsfolien der Veranstaltung Echtzeitsysteme² sowie der offiziellen eCos-Dokumentation³.

¹<http://irb.cs.tu-dortmund.de/cont/de/service/pool/info/remote.html>

²https://www4.cs.fau.de/Lehre/WS19/V_EZS/Uebung/#exercisescslides

³<http://ecos.sourceware.org/docs-latest/>

1 Aufgabenstellung

Vermerken Sie Ihre Antworten zu den Fragen der einzelnen Aufgaben an den vorgesehenen Stellen in der vorgegebenen `answers.md`. Bitte erstellen Sie, um die Abgabe durch Mergerequests zu vereinfachen, **pro Aufgabe einen eigenen Branch**. Um einen konsistenten Zustand zu gewährleisten, benutzen sie dazu bitte folgenden Befehl:

```
git fetch <vss-vorgaben> aufgabe2 && git checkout -b aufgabe2
FETCH_HEAD
```

Aufgabe 1 Datentypen und Signaturen

In dieser Aufgabe 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. Für ihre Implementierung können sie auf die Funktionalität der ISO-C-Standardbibliothek zurückgreifen.

Hinweis: Wie leider häufig in eingebetteten Systemen vorzufinden implementiert `eCos` nur eine Teilmenge der C-Standardbibliothek zur Behandlung von Fließkommazahlen⁴. Die für Sie erforderlichen Funktionen sollten jedoch vorhanden sein. Denken Sie im Zweifelsfalle über alternative Möglichkeiten der Berechnung oder Prüfung nach.

Welche Überprüfungen der Eingabedaten sollten vorgenommen werden?

Antwort:

Aufgabe 2

Vergleichen Sie die Signaturen der Funktionen `convolve_data()` und `convolve_batch()`.

Was fällt hierbei auf? Welche Auswirkungen hat das auf das jeweilige Laufzeitverhalten der Funktionen? Welche Auswirkungen könnte das auf eine statische Bestimmung

⁴Unvollständiges Errata: <https://doc.ecoscentric.com/ref/libc-iso-compliance.html#libc-iso-compliance-common-headers-math>

der schlimmstmöglichen Ausführungszeit (d.h. statische WCET-Analyse) der beiden Funktionen haben? Beantworten Sie dazu die folgende Frage: Welches Wissen muss ich über die Eingabedaten besitzen, um die Länge der Berechnung abschätzen zu können? Wovon ist die Berechnungsdauer ansonsten möglicherweise noch abhängig?

Antwort:

Aufgabe 3 Verarbeitung

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

Nachdem Sie das Filter implementiert haben, verwenden Sie es mit den entsprechenden Argumenten sowohl in `convolve_data()` und `convolve_batch()` um das vorgegebene Signal `signal_input` mit den vorgegeben Filterparametern zu falten.

Um die Ergebnisse später überprüfen und vergleichen zu können, sollen sie mittels einer Prüfsumme abgesichert werden. Hierfür soll jeweils eine Prüfsumme über die Ausgabedaten von `convolve_data()` und `convolve_batch()` generiert werden. Implementieren Sie dazu die Funktion `checksum()` und berechnen Sie die Prüfsummen mehrmals. Überprüfen Sie hier die Konsistenz des Rechenergebnisses; so sollten gleiche Eingabedaten selbstverständlich vergleichbare Ergebnisse liefern.

Welche zwei konzeptionellen Möglichkeiten sehen Sie, eine Prüfsumme für einen Datensatz (d.h. eine Folge von Zahlen/Bits) zu erstellen? Wie verhalten sich diese Varianten jeweils bezüglich Portabilität (bspw. bezüglich `REAL`) und spezifischem Fehlermuster (bspw. Einbitfehler)?

Hinweis: Bitte versuchen Sie sich hier selbst eine einfache (aber effektive) Prüfsumme zu überlegen, und implementieren Sie hier aufgrund der Komplexität keinesfalls eine der etablierten Prüfsummen wie etwa CRC32 vollständig (auch wenn dies je nach Anwendungsfall in der Praxis eine gute Wahl sein könnte).

Antwort:

http:
//mathworld.
wolfram.com/
Convolution.
html

Aufgabe 4 Fehlerbehandlung

Implementieren Sie eine Fehlerbehandlung für den Fall, dass die Prüfsummen der verschiedenen Durchläufe beziehungsweise der verschiedenen Funktionen voneinander abweichen sollten. *Welches Verhalten eignet sich hier im Fehlerfall? In welchem Zustand befindet sich der Filter im Fehlerfall?*

Antwort:

Aufgabe 5 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.

Sie sollten Ihr bisheriges Programm so strukturiert implementiert haben, dass Sie hierfür nur Änderungen in `real.h` und `real.c` vornehmen müssen. 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 Ausgabe in das Fließkomma-Format zurück transformiert werden? Wenn ja, warum?

Antwort:

Aufgabe 6

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? Begründen Sie, wie sich diese Unterschiede aus den unterschiedlichen Implementierungsvarianten ergeben.*

Antwort:

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 17.12.2020
- Fragen bitte an Peter.Ulbrich@tu-dortmund.de