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:

```
git@gitos.rrze.fau.de:i4/teaching/vezs/vezs24-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.

Hinweis: Dieses, ebenso wie die beiden folgenden Aufgabenblätter, setzen das Ihnen möglicherweise aus der Veranstaltung "Echtzeitsysteme" bereits bekannte Betriebssystem eCos ein. Für Quereinsteiger haben wir in der Vorgabe die wichtigsten, für VEZS erforderlichen Schnittstellen nochmals 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 Echtzeitsysteme[1] sowie der offiziellen eCos-Dokumentation[2].

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 git@gitos.rrze.fau.de:i4/teaching/vezs/vezs24-vorgabe.git 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

[1] `https://sys.cs.fau.de/lehre/ss24/ezs/uebung#exerciseslides`
[2] `http://ecos.sourceware.org/docs-latest/`

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.

☞ man 7p math.h

Hinweis: Wie leider häufig in eingebetteten Systemen vorzufinden, implementiert eCos nur eine Teilmenge der C-Standardbibliothek zur Behandlung von Fließkommazahlen[3]. 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 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:

[3] Unvollständiges Errata: `https://doc.ecoscentric.com/ref/libc-iso-compliance.html#libc-iso-compliance-common-headers-math`

Aufgabe 3 *Verarbeitung*

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

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

Nachdem Sie den Filter implementiert haben, verwenden Sie diesen 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 diese 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:

Aufgabe 4 *Fehlerbehandlung*

Implementieren Sie eine Fehlerbehandlung für den Fall, dass die Prüfsummen der verschiedenen Durchläufe beziehungsweise der verschiedenen Funktionen voneinander abweichen sollten. (Dies dient zur Vorbereitung auf die folgende Übung zu Fehlerinjektionsexperimenten). *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 Ausgabewerte 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:

2 Erweiterte Aufgaben

Im Rahmen der erweiterten Übung sollen Sie ein Gefühl für die Genauigkeit von Fließkommaarithmetik entwickeln. Dafür betrachten wir einen einfach Algorithmus zur Kollisionsvorhersage: Der Schnitt oder das Tangieren eines Strahls mit Winkel α und eines Kreises mit Mittelpunkt (c_1, c_2) und Radius r modelliert eine herannahende Kollision, während der Weg als frei angenommen wird, wenn Kreis und Stahl keine Schnittpunkte besitzen. Das geometrische Problem ist in Abbildung 1 mit seinen Formelzeichen dargestellt. Die Implementierung soll mittels IEEE754 Single-Precision Fließkommazahlen (d.h. Datentyp `float` erfolgen).

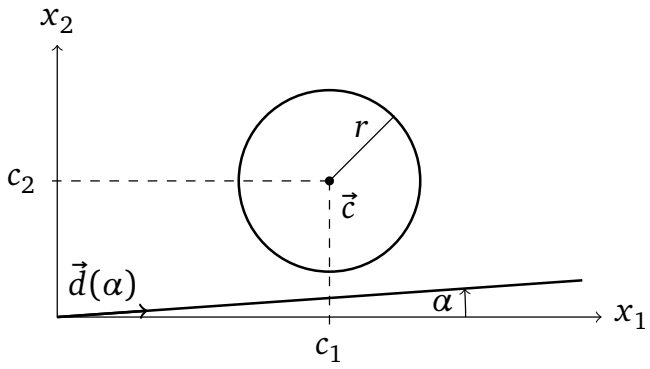

Abbildung 1: Ein Strahl aus dem Ursprung mit Richtung $\vec{d}$ schließt mit der x_1-Achse den Winkel α ein. Er schneidet den Kreis mit Mittelpunkt $\vec{c}$ und Radius r nicht.

Unter vereinfachenden Annahmen (Richtung ist Einheitsvektor, Kleinwinkelannäherung) kann durch Auswertung der Diskriminanten

$$D = (c_1 - c_2)^2\alpha^2 - 2c_1(c_1 - c_2)\alpha + (r + c_2)(r - c_2). \tag{1}$$

festgestellt werden, ob der Strahl den Kreis schneidet ($D > 0$), tangiert ($D = 0$) oder an ihm vorbei läuft ($D < 0$).

Aufgabe 7 *Vorbereitungen*

Implementieren Sie in der Datei `ext_collision_check.c` die Auswertung der Diskriminanten anhand Gleichung 1 in der Funktion `compute_discriminant_a`. Überlegen Sie sich, welche Bedingung D erfüllen muss, um Kollisionsfreiheit zu gewährleisten. Implementieren Sie die Kollisionsbedingung in der Funktion `will_collide`.

☞ `make run_ext_collision_check`

Antwort:

Aufgabe 8 *Erste Schritte*

Zunächst sollen folgende Parameter verwendet werden:

$$r = 100,\ c_1 = 1000,\ c_2 = 100,\ \alpha \in \left\{ \underbrace{\frac{\pi}{180^\circ} \cdot \frac{5^\circ}{100}}_{=:K}\, i \;\middle|\; i \in \{-100, -99, \ldots, +100\} \right\} \tag{2}$$

Skizzieren Sie das durch Gleichung 2 festgelegte Problem und folgern Sie daraus kurz und qualitativ, für welche α Sie eine Kollision erwarten. Validieren Sie dies, indem Sie die angebenen Werte in die Funktionen aus der vorherigen Aufgabe einsetzen. Verwenden Sie zur Berechnung der Winkel den vorgegebenen Faktor K. Entsprechen die Ergebnisse Ihren Erwartungen?

Antwort:

Aufgabe 9 *Eine Alternative?*

Durch Umstellen kann Gleichung 1 in folgenden Ausdruck überführt werden, der mit weniger Rechenoperationen auskommt:

$$D = ((1-\alpha)\,c_1 + \alpha\,c_2)^2 - c_1^2 - c_2^2 + r^2 \tag{3}$$

Ergänzen Sie Ihr Programm um die Funktion `evaluate_discriminant_b`, welche die Diskriminante anhand Gleichung 3 berechnen soll. Wiederholen Sie die Berechnungen aus der vorherigen Aufgabe für diese Funktion. Welche Unterschiede können Sie in den erzielten Ergebnissen feststellen?

Antwort:

Aufgabe 10 *Neue Parameter*

Ergänzen Sie die obige Skizze um folgenden, abgeänderten Parametersatz:

$$r = 100,\ c_1 = 100,\ c_2 = 100,\ \alpha \in \left\{ \underbrace{\frac{\pi}{180°} \cdot \frac{5°}{100}}_{=:K}\ i \;\middle|\; i \in \{-100, -99, \ldots, +100\} \right\} \tag{4}$$

Worin liegt der Unterschied zur vorherigen Situation? Wie wird sich das Ergebnis Ihrer Erwartung nach verändern? Führen Sie die Entscheidungsfindung auch für die Parameter aus Gleichung 4 aus. Entspricht das Ergebnis Ihren Erwartungen?

Antwort:

Aufgabe 11 *Problemanalyse I*

Für die Parameter nach Gleichung 4 gilt $r = c_1 = c_2$. Setzen Sie dies wahlweise in Gleichung 1 oder Gleichung 3 ein und eliminieren Sie möglichst viele Terme. Wie lautet Ihr Ergebnis?

Antwort:

Aufgabe 12 *Problemanalyse II*

Aus der vorherigen Aufgabe wissen Sie, dass für die Parameter aus Gleichung 4 keine sinnvollen Ergebnisse zu erwarten sind. Dies erklärt aber noch nicht die Abweichungen zwischen `compute_discriminant_a` und `compute_discriminant_b`.

Informieren Sie sich, woher die Diskrepanz stammt und erklären Sie kurz und in eigenen Worten, warum Gleichung 1 numerisch besser geeignet ist.

☞ "Catastrophic Cancellation"

Antwort:

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 08.11.2024
- Fragen bitte an `i4ezs@lists.cs.fau.de`