AUFGABE 3: ABSTRAKTE INTERPRETATION

In dieser Aufgabe werden Sie zunächst ein digitales α-β-Filter implementieren. Mehrere dieser Filter sollen als Jobs hintereinander ausgeführt werden. Für die Priorisierung der Jobs werden Sie eine erweiterte Implementierung Ihre Prioritätswartschlangen aus Aufgabe 2 wiederverwenden. Anschließend werden Sie mit Hilfe von Astrée versuchen die Korrektheit ihrer Implementierung von Warteschlange und Filter nachzuweisen.

Dieses α-β-Filter finden unter anderem Verwendung bei der Vorverarbeitung von Sensormesswerten in Flugzeugen, Automobilen etc. Im Allgemeinen ist jeder Messwert aus physikalischen Gründen durch ein Rauschen gestört, das sich als Messunsicherheit charakterisieren lässt. Sensorhardware ist jedoch in vielen Fällen fähig, häufiger zu messen, als Messwerte für die Anwendung benötigt werden. Diese zusätzlichen Sensorwerte liefern weitere Informationen über die Messgröße und können deswegen verwendet werden, um die Messunsicherheit zu reduzieren, indem aus mehreren Sensorwerten ein Schätzwert für die physikalische Größe berechnet wird.

Wenden Sie auch in dieser Aufgabe wieder die *objektbasierte Entwurfsmethode* an und verpacken Sie konzeptionell unabhängige Elemente in separate *Module*.

Hinweis: Es bietet sich an, Arbeitspakete für die einzelnen Gruppenteilnehmer zu vereinbaren. So kann beispielsweise ein Gruppenteilnehmer das Filter implementieren, während ein anderer den Allokator erstellt und der dritte Teilnehmer die Festkommabibliothek implementiert.

Aufgabenstellung

1. *Sensor-Stub:* Implementieren Sie einen Stub für einen Sensor. Dieser soll bei jedem Aufruf einen simulierten Messwert vom

Datentyp `float` liefern. Sorgen Sie dabei dafür, dass für Astrée klar ist, dass es von den auf den Übungsfolien angegebenen einschränkenden Annahmen bezüglich des konkreten Messwerts ausgehen darf. Wenn Sie möchten, können Sie hier die in der Datei `sensor.dat` enthaltenen Messwerte verwenden um Ihre Implementierung später testen zu können.

☞ volatile

☞ fscanf(3p)

2. *Warteschlange für Arbeitsaufträge:* In dieser Aufgabe sollen Sie Ihre Warteschlangenimplementierung aus der vorangegangenen Aufgabe wiederverwenden. Kopieren Sie diese hierfür in die dafür vorgesehenen Verzeichnisse unserer Vorgabe. In dieser Aufgabe soll die Warteschlange keine Ganzzahlen mehr, sondern Arbeitsaufträge verwalten. Passen Sie die Warteschlange hierzu so an, dass sie Zeiger auf Funktionen verwaltet, die keine Argumente übernehmen und keinen Rückgabewert haben.

☞ typedef Job

Die Korrektheit der Impementierung sollen Sie dieses Mal mit Hilfe von Astrée nachweisen. Astrée ist jedoch leider nicht in der Lage mit `malloc` umzugehen. Deswegen sollen Sie einen Speicherallokator für die Elemente Ihrer Warteschlange implementieren. Dieser soll Zeiger auf freie Elemente eines global angelegten Arrays zurückliefern. Welche Elemente frei sind, soll hierbei in einer Bitmaske vermerkt werden. Implementieren Sie ausserdem eine Funktion, die Elemente wieder als für die Allokation verfügbar markiert. Zeigen Sie die Korrektheit des Allokators und der erweiterten Warteschlange mit Hilfe von Astrée.

Je nach Implementierung Ihrer Warteschlange benötigen sie eventuell auch Allokatoren für Iteratoren und die Warteschlangen. Diese sind jedoch mit dem beschriebenen Muster und einer modularen Implementierung des ersten Allokators leicht zu entwickeln. Da wir in dieser Aufgabe nur jeweils eins dieser Objekte verwenden müssen können Sie diese auch statisch anlegen und auf das Entwickeln weiterer Allokatoren verzichten.

3. *Filter:* Implementieren Sie das in der Tafelübung vorgestellte α-β-Filter für den Datentyp `float` und weisen Sie dessen Korrektheit in Astrée nach.
Prinzipiell ist es ja möglich, dass das Filter sich im laufenden Betrieb in einen ungültigen Zustand begibt – *wann kann dies passieren?* Sehen Sie für diesen Fall bzw. diese Fälle eine Neuinitialisierung des Filters vor. Das Filter gilt als korrekt, wenn es kein nicht-definiertes Verhalten aufweist, nicht dauerhaft in einem ungültigen Zustand verbleibt und stabil ist.
Hinweis: Die Berechnung der *Filterparameter* für den eingeschwungenen Zustand sollten Sie mit Hilfe des Datentyps `double` vornehmen. *Wieso ist dies sinnvoll?*

4. *Integration:* Setzen Sie nun die einzelnen Bestandteile Ihres Programms zu einem Gesamtprogramm zusammen. Mehrere Sensoren sollen Messwerte liefern, die von Filtern verarbeitet und schliesslich in eine Datei geschrieben werden. Die Arbeitsaufträge, die diese Aktionen verrichten, sollen hierbei in der Initialisierungsphase erstellt, in eine Warteschlange eingereiht und dann in einer Schleife immer wieder – gemäß ihrer Priorität – abgearbeitet werden. Um das Ergebnis zu visualisieren, können Sie z. B. `gnuplot` oder `qtiplot` benutzen. Weisen Sie die Korrektheit des Gesamtprogramms mit Hilfe von Astrée nach!

Hinweise

- Bearbeitung: Gruppe mit je zwei bis drei Teilnehmern.
- Abgabezeit: die Woche vom 02.06.2015 bis zum 08.06.2015
- Fragen bitte an i4ezs@lists.informatik.uni-erlangen.de