Ostfalia
Hochschule für angewandte Wissenschaften

Institut für Fahrzeugsystem- und Servicetechnologien
IFST

Laborscript

Labor 6
Informatik II

Vorlesung : Prof. Dr. B. Lichte
Tutor : Alexander Tank
Version : 1.1 vom 4. Juni 2021

Inhaltsverzeichnis

Labor 6

Inhalt:

- Abschlussaufgabe

Ein- und Ausgabedateien:
Die Eingabedateien, die Ihr Programm verarbeiten soll finden Sind im *resources/* Verzeichnis Ihrer Programmiervorlage. Beachten Sie, dass in der Datei *resources/extract_request.txt* **nicht nur gültige Einträge/Zeilen vorhanden sind**.

Vorgehen:
Das strategische Vorgehen bei der Implementierung ist grundsätzlich Ihnen persönlich überlassen. An dieser Stelle sei aber erwähnt, dass es sinnvoll sein kann, die Funktion in der gegebenen Reihenfolge zu implementieren und ihre Implementierungen stets in kleinen Schritten zu überprüfen.

Hinweise:

1. Für ein besseres Verständnis der Aufgabe beachten Sie unbedingt die nachfolgenden Informationen zum Aufbau als auch das Aktivitätsdiagramm in Abbildung 6.1.
2. Verwenden Sie in Ihrem Programm **relative Pfade** zu den Dateien.
3. Beachten Sie, dass Ihr Programm für eine **beliebige Anzahl an Zeilen** in *resources/signal.txt* und *resources/extract_request.txt* funktionieren muss.
4. Beachten Sie die **CodingConventions**.
5. Zur Abgabe der Aufgabe exportieren Sie bitte die Datei wie im Labor gezeigt als ZIP-Archive-File. Bitte beachten Sie die Namensgebung der exportierten Datei:

   ```
   SS2021_Name_Vorname_Matrikelnummer.zip
   ```

6. Laden Sie das erzeugte Archive-File in den für Sie vorgesehenen Ordner im Stud.IP hoch.

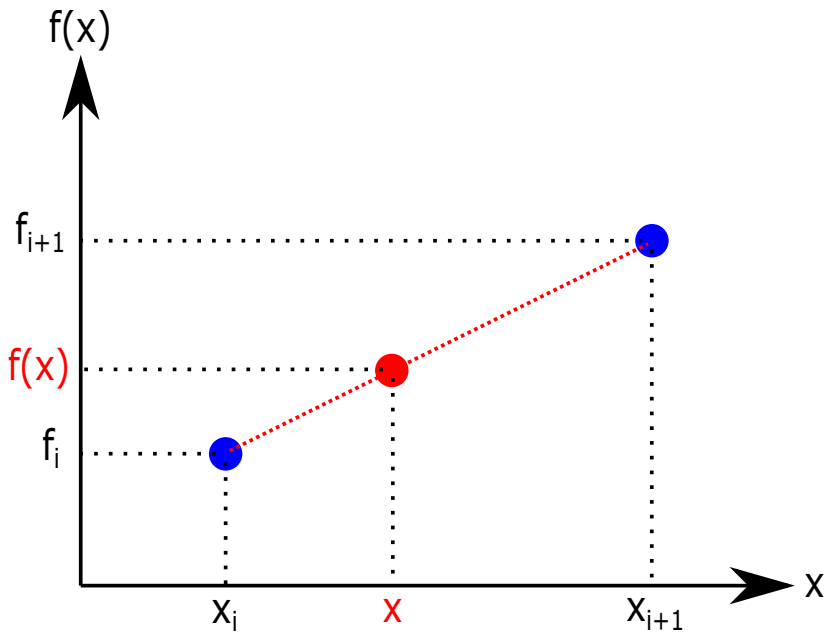
Viel Erfolg!

Abschlussaufgabe

In der Abschlussaufgabe sollen Sie eine diskrete Messreihe in Abhängigkeit von der Zeit aus einer Datei einlesen, um anschließend beliebige Werte aus dieser Messreihe auf Basis einer Zeitangabe extrahieren zu können. Die einzulesende Messreihe finden Sie in der Datei *resources/signal.txt*. Des Weiteren finden Sie eine Beispieldatei für die zu extrahierenden Werte in der Datei *resources/extract_request.txt*. Die Messreihe wurde mit einer konstanten Abtastrate von 1s aufgezeichnet. Für die Extraktion sollen zwei Methoden realisiert werden:

- `Hold and Sample`: Wenn der angeforderte Zeitwert zwischen zwei Messwerten liegt, soll der Wert des zeitlichen Vorgängers zurückgegeben werden.
- `Lineare Interpolation`: Wenn der angeforderte Zeitwert zwischen zwei Messwerten liegt, soll linear zwischen beiden Werten interpoliert werden. Die Berechnung der linearen Interpolation funktioniert wie folgt:

$$f(x) = f_i + (x - x_i) \cdot \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$$



Aufbau:
Um die anschließende Bewertung zu vereinfachen und vergleichbarer zu machen, wird eine gewisse Grundstruktur des Programms in Form von Funktionen vorgegeben. Darüber hinaus können somit auch einfacher Teilpunkte vergeben werden, wenn das Gesamtprogramm nicht korrekt funktioniert, einzelne Programmteile jedoch korrekt implementiert sind. Die Funktionalität Ihres Programms soll daher auf die folgenden Funktionen aufgeteilt werden.

- `int determineNumberOfLines(FILE *pSampleInputFile);`
  Der Funktion soll ein Filepointer `pSampleInputFile` übergeben werden, über den die Funktion die Eingabedatei einließt und analysiert, wie viele Zeilen/Datensätze darin gespeichert sind. Dieser Wert soll als Ergebnis anschließend zurückgegeben werden.

- `void readInputData(FILE *pSampleInputFile, Sample_t *pSamples, int iNumberOfSamples);`
  Der Funktion soll ein Filepointer `pSampleInputFile` übergeben werden, aus der die Funktion insgesamt eine Anzahl von `iNumberOfSamples` Datensätzen einließt und in ein Feld `pSamples` des dafür zu erstellenden Datentyps `Sample_t` abspeichert (siehe Aktivitätsdiagramm).

- `double holdExtraction(Sample_t *pSamples, double dTime);`
  Diese Funktion realisiert die Methode `Hold and Sample`. Sie bekommt die Messreihe als Pointer und den zu berechnenden Zeitwert übergeben. Als Rückgabewert wird der entsprechende Wert des Vorgängers zurückgegeben.

- `double linearExtraction(Sample_t *pSamples, double dTime);`
  Diese Funktion realisiert die Methode `Lineare Interpolation`. Sie bekommt die Messreihe als Pointer und den zu berechnenden Zeitwert übergeben. Als Rückgabewert wird der entsprechende Wert der linearen Interpolation zurückgegeben.

- `void performExtraction(Sample_t *pSamples, FILE *pExtractRequestFile, FILE *pExtractResultFile);`
  Nachdem die Messreihe in ein Array vom Typ `Sample_t` eingelesen wurde, soll das Array an die Funktion `performExtraction` übergeben werden. In dieser Funktion sollen aus der Datei *pExtractRequestFile* zeilenweise die zu extrahierenden Zeitpunkte und die zu verwendende Methode ausgelesen werden. `method=0` steht für `Hold and Sample` und `method=1` steht für `Lineare Interpolation`. Verwenden Sie für die Berechnung der Methoden die zuvor erläuterten Funktionen `holdExtraction` und `linearExtraction`. Die Ergebnisse sollen wiederum zeilenweise in die Datei *pExtractResultFile* geschrieben werden. Achten Sie bei der zeilenweisen Betrachtung explizit auf die korrekte Syntax der Zeile: „f(*Zeit*),method=*Methode*“. Falls diese abweicht, soll an die entsprechende Zeile in der Datei *pExtractResultFile* „Syntax nicht korrekt!“ geschrieben werden.

Diese Funktionen sollen aus der `main()`-Funktion nach folgendem Schema aufgerufen werden:

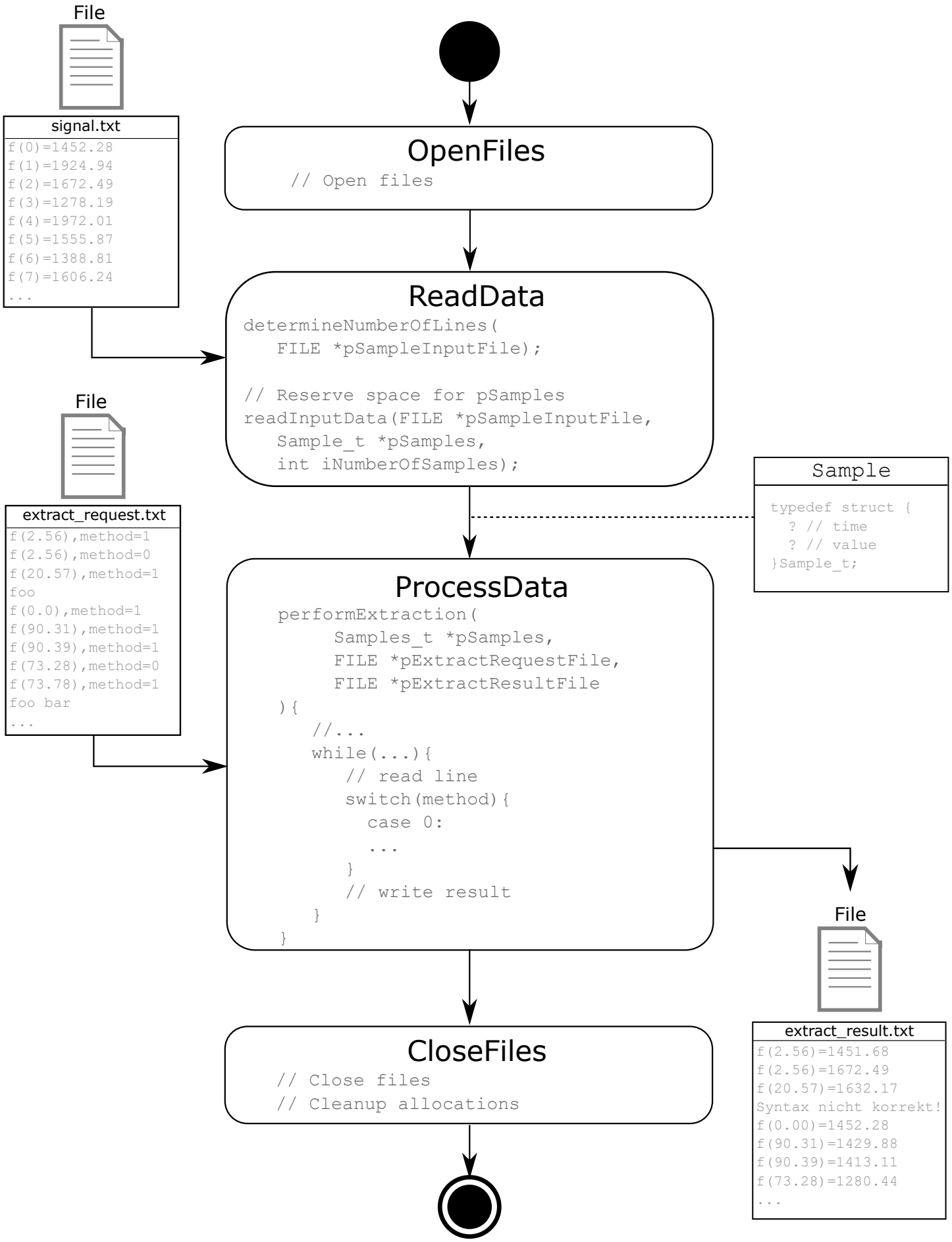


Abbildung 6.1: Aktivitätsdiagramm