Sprachelemente

I. Einleitung

In diesem Praktikum werden Sie Eingaben von der Kommandozeile einlesen und verarbeiten sowie erhaltene Ergebnisse ausgeben. Es dient als Vorbereitung für das Praktikum in der folgenden Woche, welches auf den Inhalten dieses Praktikums aufbauen wird.
Die Aufgaben in diesem – und in den weiteren Praktika – sind bewusst so angelegt, dass die nötige Vorbereitungs- und Bearbeitungszeit über die reine Präsenzzeit hinausgeht. Nutzen Sie die Zeit im Praktikumstermin in der Hochschule, z.B. um Fragen zu stellen, Hilfestellung zu bekommen und Schwierigkeiten zu besprechen. Bereiten Sie sich dazu im Selbststudium – am besten in Ihrer Zweiergruppe – auf den Praktikumstermin vor und arbeiten Sie anschließend weiter.

II. Vorbereitung

Lesen Sie noch einmal die Arbeitsschritte nach, die Sie im Praktikum zum Kursthema „Einführung“ durchgeführt haben. Sie sollten diese zumindest stichpunkthaft im Kopf haben.

III. Aufgaben – Teil 1: Summe, Minimum, Maximum

In diesem Teil implementieren Sie eine Konsolenanwendung, die so lange die Eingabe einzelner Zahlen akzeptiert bis *Strg + D* (unter Windows ggf. *Strg + Z* und *Enter* bzw. *Return*) gedrückt wird. Die Eingabe einer einzelnen Zahl wird dabei jeweils mit *Return* oder *Enter* abgeschlossen. Nach der Eingabe aller gewünschten Zahlen wird die Summe der Zahlen sowie das Minimum und das Maximum ausgegeben.
Da es zum Testen mühsam ist, immer wieder mehrere Zahlen von Hand einzugeben, nutzen Sie die Möglichkeiten der Eingabeaufforderung, die Zahlen aus einer Eingabedatei an Ihre Anwendung weiterzuleiten. Zusätzlich nutzen Sie die Möglichkeit der Eingabeaufforderung, die Ausgabe Ihrer Konsolenanwendung in eine Datei umzuleiten.

1. **Repository anlegen**

   Legen Sie in Ihrem Ordner für die GIP1-Programme ein neues Repository als Unterverzeichnis mit einem geeigneten Namen an.

2. **Hello 17**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   (a) Legen Sie in dem neuen Repository eine neue Datei `main.c` an.

   (b) Implementieren Sie in dieser Datei die `main`-Funktion. Diese soll mittels `printf` lediglich die Zahl `17` an der Konsole ausgeben.

   (c) Erzeugen Sie Ihre Konsolenanwendung im Unterverzeichnis `build` und beheben Sie ggf. auftretende Fehler.

   (d) Testen Sie Ihre Konsolenanwendung und beheben Sie ggf. auftretende Fehler.

   (e) Legen Sie die durchgeführten Änderungen im Repository ab. Vergeben Sie eine aussagekräftige Commit-Message.

3. **Ein- und Ausgabe**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   (a) Implementieren Sie in der `main`-Funktion die Tastatureingabe einer Ganzzahl.

   (b) Ändern Sie die Ausgabe in die Konsole so, dass die zuvor eingegebene Zahl wieder ausgegeben wird.

   (c) Erzeugen Sie Ihre Konsolenanwendung im Unterverzeichnis `build` und beheben Sie ggf. auftretende Fehler.

   (d) Testen Sie Ihre Konsolenanwendung und beheben Sie ggf. auftretende Fehler.

   (e) Legen Sie die durchgeführten Änderungen im Repository ab. Vergeben Sie eine aussagekräftige Commit-Message.

4. **Eingabe mehrerer Zahlen ermöglichen**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   Ab dieser Aufgabe wird das Erzeugen der Konsolenanwendung, die Fehlerbehebung und das Ablegen im Repository nicht mehr explizit erwähnt. Führen Sie diese Schritte selbsständig an sinnvollen Stellen durch, während Sie die Aufgaben bearbeiten.

   Nun soll Ihre Konsolenanwendung die Eingabe mehrerer Zahlen akzeptieren. Die Eingabe einer einzelnen Zahl wird nach wie vor mit *Enter* oder *Return* abgeschlossen. Nach wie vor soll nach der Eingabe einer einzelnen Zahl die Zahl selbst ausgegeben werden. Das Programm soll beendet werden, wenn *Strg + D* (bzw. *Strg + Z*) gedrückt wird. Dies signalisiert der Standardeingabe, dass das Ende der Eingabe erreicht/gewünscht ist.

   Erweitern Sie Ihr Programm um die beschriebene Funktionalität.

   Hinweis zur Implementierung: Die Funktion `scanf` gibt Ihnen eine Ganzzahl zurück. Diese zurückgegebene Ganzzahl hat den Wert `EOF`, wenn während der Eingabe *Strg + D* (bzw. *Strg + Z*)gedrückt wurde.

5. **Summe der eingegebenen Zahlen**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   (a) Erweitern Sie Ihr Programm so, dass es noch während die Eingabe der Zahlen läuft, die Summe der bisher eingegeben Zahlen ermittelt.

   (b) Enfernen Sie die Ausgabe der jeweils aktuell eingegebenen Zahl. Sobald die Eingabe aller Zahlen beendet ist, soll stattdessen die ermittelte Summe ausgegeben werden. Die Eingabe

   ```
   17
   3
   9
   8
   ```

   soll folgende Ausgabe erzeugen:

   ```
   sum: 37
   ```

6. **Verarbeiten einer Eingabedatei mit Zahlen**

   Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

   Die Eingabeaufforderung erlaubt es, den Inhalt ganzer Dateien als Eingabe in ein Programm zu leiten. Hier im Praktikum können Sie sich dadurch beim Testen Ihrer Programme das wiederholte, zeitaufwendige Eintippen von Eingabedaten sehr viel angenehmer gestalten.

   Falls ein Programm `ANWENDUNG` also beispielsweise Daten von der Standardeingabe verarbeitet und Sie ihm hierzu die Daten aus der Datei `input.txt` zur Verfügung stellen möchten, dann können Sie das folgendermaßen tun:

   ```
   $ ANWENDUNG < input.txt
   ```

   (a) Legen Sie eine Textdatei (z. B. `input.txt`) mit folgendem Inhalt an.

   ```
   1
   2
   3
   ```

(b) Um den Inhalt dieser Datei vor Verwendung zu überprüfen, können Sie den Inhalt dieser Datei an der Kommandozeile ausgeben:
Linux / macOS:

```
$ cat input.txt
```

Windows:

```
> type input.txt
```

Bei $ und > am Zeilenanfang handelt es sich jeweils um den sogenannten *prompt character*. Dieser macht deutlich, dass es sich bei der entsprechenden Zeile um eine Eingabeaufforderung handelt. Der *prompt character* wird Ihnen in der von Ihnen verwendeten Konsole ebenfalls angezeigt.

(c) Geben Sie den Inhalt der obigen Textdatei als Eingabe an Ihre Konsolenanwendung weiter.

7. Umleiten der Ausgabe in eine Datei

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

Die Eingabeaufforderung erlaubt es, die Ausgabe in eine Datei umzuleiten. Dies geschieht mittels >. Die Ausgabe des Befehls `ANWENDUNG` wird beispielsweise folgendermaßen in die Datei `output.txt` umgeleitet:

```
$ ANWENDUNG > output.txt
```

(a) Geben Sie den Inhalt der in der vorherigen Aufgabe angelegten Textdatei `input.txt` an der Konsole aus. Leiten Sie die Ausgabe anschließend in eine Datei um (z. B. `output.txt`).
Überprüfen Sie den Inhalt der Datei `output.txt`. Es sollte eine exakte Kopie von `input.txt` sein.

(b) Geben Sie den Inhalt der Eingabedatei `input.txt` an Ihre Konsolenanwendung weiter und leiten Sie die Ausgabe dieser Konsolenanwendung in eine Ausgabedatei um.

8. Minimum und Maximum der eingegebenen Zahlen

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter.

(a) Erweitern Sie Ihr Programm so, dass es noch während die Eingabe der Zahlen läuft, das Maximum der bisher eingegeben Zahlen ermittelt.

(b) Erweitern Sie Ihr Programm so, dass es noch während die Eingabe der Zahlen läuft, das Minimum der bisher eingegeben Zahlen ermittelt.

(c) Sobald die Eingabe aller Zahlen beendet ist, soll zusätzlich zur Summe das Minimum und das Maximum der eingegeben Zahlen ausgegeben werden. Die Eingabe

```
17
3
9
8
```

soll folgende Ausgabe erzeugen:

```
sum: 37
max: 17
min: 3
```

(d) Lagern Sie die Ermittlung des aktuellen Minimums und Maximums jeweils in eine eigene Funktion aus.

IV. Aufgaben – Teil 2: Komplexere Eingabedaten

9. Temperatur-Daten

Legen Sie für diese Aufgabe einen neuen Ordner auf Ihrer Festplatte mit einem neuen Git-Repository an.

(a) Implementieren Sie eine Konsolenanwendung, die nur eine einzelne Ganzzahl einliest und anschließend alle Ganzzahlen von 1 fortlaufend bis zu der eingegebenen Zahl ausgibt.

(b) Rufen Sie in der *cppreference* (https://cppreference.com) die Dokumentation der C-Funktion `scanf` auf. Lesen Sie darin die ersten beiden Punkte zum *format string* (einer der Parameter).

(c) Implementieren Sie anstelle der Ausgabe der Ganzzahlen eine wiederholte Eingabe im Format:

```
GANZZAHL,FLIESSKOMMAZAHL
```

Dabei steht `GANZZAHL` für eine Ganzzahl und `FLIESSKOMMAZAHL` für eine Fließkommazahl.
Nach jeder Eingabe soll eine Ausgabe in folgendem Format erfolgen:

```
GANZZAHL : FLIESSKOMMAZAHL
```

Konkret soll Ihr Programm für diese Eingabe

```
1
2,3.14
```

folgende Ausgabe erzeugen

```
2 : 3.14
```

sowie für diese Eingabe

```
3
2,4.2
6,8.4
10,12.6
```

folgende Ausgabe:

```
2 : 4.2
6 : 8.4
10 : 12.6
```

Ihr Programm soll jeweils nach der gezeigten Ausgabe ohne Fehler enden.

(d) Erweitern Sie Ihr Programm nun so, dass während des Einlesens der Daten keine Ausgaben stattfinden. Nachdem alle Daten eingelesen sind, soll Ihr Programm die erste und die letzte eingelesene Ganzzahl ausgeben sowie das Minimum, das Maximum und das arithmetische Mittel aller eingelesenen Fließkommazahlen. Stellen Sie mit obigen Eingaben sicher, dass Ihr Programm korrekt funktioniert.

(e) Laden Sie nun die bereitgestellte Eingabedatei aus dem Moodle-Kurs oder den Online-Materialien zu diesem Kurs[1]. Diese Datei liegt im oben beschriebenen Format vor. Lassen Sie Ihr Programm die Datei einlesen und notieren Sie die Ausgabe. Beheben Sie gegebenenfalls auftretende Fehler.

[1] `https://vc.w-hs.de/teaching/gip1-2023/020/p`