Aufgabe 1 – Einführung in DDD

Quelle: http://www.gnu.org/software/ddd/manual/html_mono/ddd.html#Sample%20Session

Programm ‚sample' nimmt in der Kommandozeile Ganzzahlen als Argumente entgegen und gibt sie sortiert aus:

```
$ ./sample 8 7 5 4 1
1 4 5 7 8
```

Führen Sie das Programm mit folgenden Argumenten aus:
1) `8 7 5`
2) `100000 2000000 3 5`

Zunächst muss das Programm mit Debug-Informationen kompiliert werden. Dies geschieht über Angabe des Flags '-g' an den Compiler: '`gcc -g -o sample sample.c`'
Dazu starten Sie den Debugger ddd:

```
$ ddd sample
```

Wenn DDD startet, sehen Sie den Quellcode von sample. Setzen Sie einen Breakpoint in Zeile 31 mit einem Doppelklick auf den Zeilenanfang oder mit rechter Maustaste und "Set Breakpoint" in der Toolbar.
Führen Sie dann mit ‚Program => Run' im Menü das Programm aus. Es öffnet sich ein Dialog, in dem Sie zur Eingabe von Argumenten für das Programm aufgefordert werden.
Geben Sie Argumente ein, die einen Fehler auslösen, und
drücken Sie auf "Run".

Die aktuelle Position im Quelltext wird Ihnen während der Ausführung mit einem grünen Pfeil markiert.

Jetzt können Sie den Wert einer Variablen untersuchen; bewegen Sie dafür den Zeiger über diese. Alternativ kann der Wert über einen Doppelklick auf die Variable im Data-Fenster angezeigt werden.

Geben Sie unten in der Konsole "`print a[0]`" ein, um den aktuellen Wert von a[0] ausgeben zu lassen.

Eine alternative Möglichkeit bietet das Eingabefenster in der Toolbar. Tippen Sie dort "`a[0]@(argc - 1)`" ein. (argc ist die Variable argc aus der Main-Funktion. In C bekommt diese die Argumentanzahl als argc übergeben.)
Drücken Sie danach den Button "Print". In der Konsole werden nun die aktuellen Werte für alle Elemente von a angezeigt.
Alternativ kommt die beständige, sich selbstständig aktualisierende Ansicht im Data Window, wenn Sie den Button "Display" drücken.
Mit der Maus können Sie den dort entstandenen Eintrag markieren, bewegen und mit dem "Rotate"-Button horizontal oder vertikal ausrichten.

<u>Zuweisung der Argumente in den Array.</u>
Klicken Sie zwei Mal auf "Next". Sie sehen, wie schrittweise die Argumente auf die Elemente des Arrays zugewiesen werden.
Um die Schleife zu überspringen, klicken Sie auf "Until", bis Sie die Schleife verlassen haben.
Ab jetzt beinhaltet a alle Argumente, die Sie übergeben haben. Der Ausführungszustand steht jetzt auf `shell_sort(a, argc);`

Drücken Sie wieder auf "Next", um die Sortierfunktion aufzurufen.
Nach der Berechnung legt shell_sort seine Ergebnisse ebenfalls im Array a ab.
Wie Sie erkennen können, scheint der Fehler in der Funktion shell_sort aufzutreten.

Platzieren Sie nun erst einen Breakpoint an den Aufruf von shell_sort und starten Sie das Programm neu. Sie können den Breakpoint einfach mit der Maus verschieben, oder den alten
löschen (rechte Maustaste - "Delete Breakpoint" auf dem Stopzeichen, oder in die Zeile klicken und in der Toolbar "Delete" anwählen). Zum Neustart mit gleichen Argumenten
drücken Sie entweder auf den "Run"-Button im Command Tool oder über das Menü: Program => Run Again.

Die neue Ausführung wird in Zeile 35 beim Aufruf von shell_sort stehen bleiben. Drücken Sie nun "Step" im Command Tool, um die Funktion zu betreten.
Die Ausführungsanzeige ist in die Funktion gesprungen und die Konsole zeigt an, dass der Debugger die Funktion betreten hat.

```
(gdb) step
    shell_sort (a=0x8049878, size=6) at sample.c:9
```

Die main Funktion ist auf dem Stack plaziert, was Sie mit der Konsoleneingabe 'stack frame display' oder über das Menü ("Status" => "Backtrace") sehen können. Merken Sie, dass beim Wechseln des Kontextes ebenfalls die Sichtbarkeit von Variablen sich ändert.
Daher wird im Data Window a nicht mehr angezeigt. (In diesem Fall ist a identisch. Nicht identisch ist hier bspw. i; da sowohl eine Anzeige der "alten" als auch der "neuen" Variable verwirren können, wird bei jedem neuen Funktionsaufruf alle vorherigen angezeigten Variablen versteckt.)

Überprüfen Sie, ob die an shell_sort übergebenen Argumente korrekt sind.
Dafür geben Sie '`a[0]@size`' in das Eingabefenster ein und drücken Sie auf "Print" oder "Display".
Wie Sie sehen, ist der letzte Eintrag von a unsinnig. Wie kommt dieser Fehler zustande?
size ist um 1 grösser als die Anzahl der übergebenen Argumente an das Programm. Da in C das erste Argument, das main() übergeben bekommt, der Name der Programmdatei ist, wird beim Einlesen eine Iteration zu viel gemacht. Dabei wird der Wert der Speicherzelle direkt hinter dem letzten Eintrag von argc ebenfalls als "Argument" angenommen und nach dem Parsen in a gespeichert.

Um zu sehen, ob dies der Grund für das Problem ist, ändern Sie den Wert von size, um nur die relevanten Felder zu berücksichtigen.
Verringern Sie daher size um 1. Markieren Sie dazu den Variablennamen im Quellcode und klicken auf "Set".
In dem sich öffnenden Dialog steht der aktuelle Wert. Ändern Sie diesen, indem Sie die um 1 kleinere Zahl eingeben, oder der angezeigten Zahl ein '-1' anfügen. Drücken Sie dann auf "OK". Über den Eintrag "Finish" im Command Tool lassen Sie das Programm laufen, bis es wieder in den nächsthöheren Stackframe zurückkehrt.

Nun sehen Sie wieder die alten beobachteten Variablen im Data Window. a enthält nun die richtigen Ergebnisse.
Lassen Sie das Programm mit "Cont" im Command Tool bis zum Ende laufen. Die Ausgabe in der Konsole sollte nun die richtigen Werte enthalten.

Korrigieren Sie nun den Quellcode, der zu dem Fehler geführt hat. Überprüfen Sie, ob Ihr Programm nun fehlerfrei funktioniert.

Aufgabe 2 – Endlosschleife

Wenn Sie a2 ausführen, wird es sich nie beenden, da es sich irgendwo in einer Endlosschleife verfängt. Bitte finden Sie die versteckten Fehler mit dem Debugger und korrigieren Sie das Programm.

Aufgabe 3 – Fehlerkorrektur

Finden Sie die Fehler in Dateien a3_1.c, a3_2.c, a3_3.c und korrigieren Sie die Programme, ohne dabei größere Änderungen vorzunehmen.

Tipps:
a3_1: 2 Fehler
a3_2: 1 Fehler
a3_3: 3 Fehler

Aufgabe 4 – Zeiger, Adressen und so

1. Kompilieren Sie a4.c und starten Sie es mit ddd.
Schauen Sie sich an, wie die verschiedenen Parameter übergeben werden.

Parameter	**By Value**	**By Reference**
in		
* in		
si		
* si		
x		
* x		
y		
* y		
ein		
* ein		

2. Was macht die Funktion w(x,y)?

3. Funktion s().
- **i)** Was ist die Variable m, und welchen Wert hat sie?
- **ii)** Wie entwickeln sich die Werte von st und e?
- **iii)** Was für ein Algorithmus liegt der Funktion s zu Grunde?

4. Funktion m().
- **i)** Was wird an die Funktion s() weiter gegeben?
- **ii)** Was passiert nach der Speicherfreigabe?

5. Korrigieren Sie das Programm.