Universität Bremen

Korrekte Software: Grundlagen und Methoden SoSe 17

Serge Autexier
Christoph Lüth

8. Übungsblatt

Ausgabe: 15.06.17
Abgabe: 22.06.17

8.1 *Verifikation mit Strukturen und Pointern* *10 Punkte*

Gegeben sei folgende Datenstruktur für Vektoren: `struct Vektor { int x; int y; };`

Vektoren haben eine Norm, die sich aus der Quadratwurzel der Summe der Quadrate der beiden Komponenten x und y ergibt.

1. Implementieren Sie folgende C Funktion

   `struct Vektor *LaengsterVektor(struct Vektor *a, struct Vektor *b),`

   die zwei Vektoren bzgl. ihrer Normen vergleicht und jenen zurückliefert, der die größere Norm hat.
   Hinweis: Implementieren Sie die Funktion ohne die Quadratwurzel zu nutzen.
2. Spezifizieren Sie die Korrektheit der Funktion mit entsprechenden Vor- und Nachbedingungen.
3. Annotieren Sie den Programmteil der Funktion in der von der Vorlesung gewohnten Notation des Hoare-Kalküls zur Verifikation der Korrektheit.

8.2 *Verifikation mit Arrays* *10 Punkte*

1. Implementieren Sie folgende C Funktion `void sort(int size, int x[]),`
   die ein Feld gegebener Größe sortiert. Verwenden Sie hierfür ein einfaches Sortierverfahren, etwa "Sortieren durch Austauschen/Bubblesort".
2. Spezifizieren Sie die Korrektheit der Funktion mit entsprechenden Vor- und Nachbedingungen.
3. Annotieren Sie das Programmm mit Invarianten und berechnen Sie die Verifikationsbedingungen. Dabei können Sie auf folgende Spezifikationen von mathematischen Funktionen zurückgreifen:

```
/** int count(int i, int size, int x[]) */
/** assert \forall int i; \forall int x[]; count(i,0,x) = 0; */
/** assert \forall int i; \forall int size; \forall int x[]; size > 0 ⟶ count(i,size,x) ==
    (count(i,size−1,x) + (x[size−1]== i)) */

/** int permutation(int size, int x[], int y[]); */
/** assert \forall int size; int x[], int y[]; (size ≥ 0 && \forall int i; count(i,size,x) ==
    count(i,size, y)) ⟶ permutation(size, x,y) == 1 */
/** asssert \forall int size; int x[], int y[]; (size ≥ 0 && !(\forall int i; count(i,size,x)
   == count(i,size, y))) ⟶ permutation(size, x,y) == 0 */

/** int sorted(int size, int x[]) */
/** assert \forall int x[]; sorted(0, x) == 1 && sorted(1, x) == 1 */
/** assert \forall int size; int x[]; size > 1 && x[size−2] ≤ x[size−1] ⟶ sorted(size,x) =
    sorted(size−1,x) */
/** assert \forall int size; int x[]; size > 1 && !(x[size−2] ≤ x[size−1]) ⟶ sorted(size,x) =
     0 */
```