

Software-Reengineering

WS 2006/07
G. Vogel

7. Übungsblatt

Dieses Übungsblatt soll den Vorlesungsstoff zum Program-Slicing, zur Klonerkennung und zu Metriken wiederholen und vertiefen. Es wird am 10./11.1.2007 besprochen.

Aufgabe 7.1 *Summary-Edges*

```
1 function Select(A, B, C, D : in Integer; L : in Natural)
2   return Integer
3 is
4 begin
5   if L=0 then
6     return A;
7   else
8     return Select(B, C, D, -1, L-1);
9   end if;
10 end Select;
```

Berechnen Sie für die Funktion `Select` die Summary-Edges.

Hinweis: Behandeln sie die `return`-Statements wie Zuweisungen an eine lokale Variable `Result`.

Aufgabe 7.2 *Klonerkennung nach Baker*

Finden Sie mit Hilfe des Verfahrens von Baker geklonte Fragmente im folgenden C-Programm:

```
1
2 for (; c > 0; c--) {
3   *p = *p + 1;
4 }
5
6 for (; i > 0; i--) {
7   *s = *s + 1;
8   *p = *p * 1;
9 }
```

Aufgabe 7.3 *Hash-Funktion für ASTs*

Skizzieren Sie eine Hash-Funktion, die abstrakte Syntaxbäume entsprechend der Grammatik für Ausdrücke aus Aufgabe 2 auf Blatt 2 mit möglichst wenig Kollisionen auf die Menge der natürlichen Zahlen abbildet. Ihre Hash-Funktion

sollte von Bezeichnern innerhalb der abzubildenden abstrakten Syntaxbäumen abstrahieren. So sollten also etwa die beiden folgenden Fragmente denselben Hash-Index haben:

```
p + 1
a + 1
```

Aufgabe 7.4 *Software-Metriken*

Bestimmen Sie die Metriken LOC, McCabe und Halstead-Volumen für die Sortierfunktionen *lqksort/qksort* und *bubbleSort* in den folgenden beiden Programmen:

```
1 // original: http://www.d.umn.edu/~gshute/C/examples/quicksort.C.html
2 #include <stdio.h>
3
4 int A[] = { 99, 43, 22, 17, 57, 32, 43, 19, 26, 48, 87, 12, 75, 0 };
5 const int numEntries = sizeof(A)/sizeof(int);
6
7 void lqksort(int ilo, int ihi) {
8     int pivot;           // pivot value for partitioning array
9     int ulo, uhi;       // indices at ends of unpartitioned region
10    int ieq;             // least index of array entry with value equal to pivot
11    int tempEntry;      // temporary entry used for swapping
12
13    if (ilo >= ihi) {
14        return;
15    }
16    // Select a pivot value.
17    pivot = A[(ilo + ihi)/2];
18    // Initialize ends of unpartitioned region and least index of entry
19    // with value equal to pivot.
20    ieq = ulo = ilo;
21    uhi = ihi;
22    // While the unpartitioned region is not empty, try to reduce its size.
23    while (ulo <= uhi) {
24        if (A[uhi] > pivot) {
25            // Here, we can reduce the size of the unpartitioned region and
26            // try again.
27            uhi--;
28        } else {
29            // Here, A[uhi] <= pivot, so swap entries at indices ulo and
30            // uhi.
31            tempEntry = A[ulo];
32            A[ulo] = A[uhi];
33            A[uhi] = tempEntry;
34            // After the swap, A[ulo] <= pivot.
35            if (A[ulo] < pivot) {
36                // Swap entries at indices ieq and ulo.
37                tempEntry = A[ieq];
38                A[ieq] = A[ulo];
39                A[ulo] = tempEntry;
40                // After the swap, A[ieq] < pivot, so we need to change
41                // ieq.
42                ieq++;
43                // We also need to change ulo, but we also need to do
```

