*Prof. Dr. Wolfgang Effelsberg*
*Dirk Stegemann* Mannheim, den 20.06.2005

Vorlesung **Programmierkurs 2**, Sommersemester 2005

# Übungsblatt 10

**Abgabe** spätestens am Montag, 27. Juni 2005, um 13.45 Uhr.

## Aufgabe 1: Ungerichtete Graphen

(8 Punkte, Abgabedatei `graph.c`)

In dieser Aufgabe betrachten wir ungerichtete Graphen $G = (V, E)$ mit einer Knotenmenge $V = \{v_1, \ldots, v_n\}$ und einer Kantenmenge $E \subseteq \{(v_i, v_j) | v_i, v_j \in V\}$. Da der Graph ungerichtet ist, gilt $(v_i, v_j) \in E \Rightarrow (v_j, v_i) \in E$. Jeder Knoten $u$ soll mit Hilfe der folgenden Datenstruktur verwaltet werden:

```
typedef struct graphknoten{

  void* inhalt;
  knoten* nachbarn;
} vertex;
```

Dabei bezeichnet `nachbarn` einen Zeiger auf eine generische verkettete Liste (vgl. Aufgabe 2 auf Übungsblatt 8), in der alle Nachbarn von $u$ gespeichert werden, d.h. alle Knoten $v$, für die eine Kante $(u, v)$ existiert.

Implementiere die folgenden C-Funktionen zur Verwaltung von ungerichteten Graphen:

- `vertex* neuerGraphknoten(void* inhalt)` erzeugt einen neuen Graphknoten mit dem übergebenen Inhalt. Die Funktion gibt `NULL` zurück, falls der Knoten nicht erzeugt werden konnte.
- `void neueKante(vertex* a, vertex* b)` erzeugt eine neue Kante zwischen den Graphknoten $a$ und $b$.
- `void loescheGraphknoten(vertex* a)` löscht alle Kanten zu den Nachbarn von $a$ und gibt den von $a$ belegten Speicherplatz frei.
- `void loescheKante(vertex* a, vertex* b)` löscht die Kante zwischen den Knoten $a$ und $b$.

Verwende für die Listenoperationen deine Lösung von Übungsblatt 8 oder die entsprechenden Lösungshinweise.

## Aufgabe 2: Erweiterte Binärbäume

(7 Punkte, Abgabedatei `genbaum2.c`)

Erweitere deine Implementation der generischen Binärbäume von Übungsblatt 9 um eine Funktion

```
void loescheInhalt(knoten *baum, void* inhalt, int (*comp)(void* inhalt1, void*
                                    inhalt2)),
```

die alle Knoten mit dem gegebenen `inhalt` aus dem Binärbaum `baum` löscht und dabei die Ordnung der Knoteninhalte im `baum` beibehält.

## Aufgabe 3: Adressierungsarten

(5 Punkte, Abgabe nur schriftlich)

Gegeben sei die folgende Speicher- und Registerbelegung eines MSP430-Mikroprozessors (alle Zahlen in Hexadezimaldarstellung).

| | | | |
|---|---|---|---|
| R4: | 014a | 014c: | 0123 |
| R5: | 0140 | 014a: | 0222 |
| R6: | 0130 | 0148: | 0345 |
| | | 0146: | 0444 |
| | | 0144: | 0567 |
| | | 0142: | 0666 |
| | | 0140: | 0789 |

Welche Belegung haben Register und Speicher nach Ausführung der folgenden Befehle (mit Begründung)?

(i)
```
MOV R4,0(R5)
```

(ii)
```
MOV @R4, R5
```

(iii)
```
MOV #0144h, R5
MOV @R5+, R4
```

(iv)
```
MOV 4(R5), &0144h
```

(v)
```
MOV 4(R5), R6
MOV &0146h, -6(R4)
MOV R6,6(R5)
```

Bearbeite jede Teilaufgabe unabhängig von den anderen, d.h. gehe für jede Teilaufgabe wieder von den ursprünglichen Speicher- und Registerinhalten aus.