TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät Informatik • Institut für Theoretische Informatik • Lehrstuhl Grundlagen der Programmierung
M. Sc. **T. Stüber** / Dr.-Ing. **L. Rüdiger**

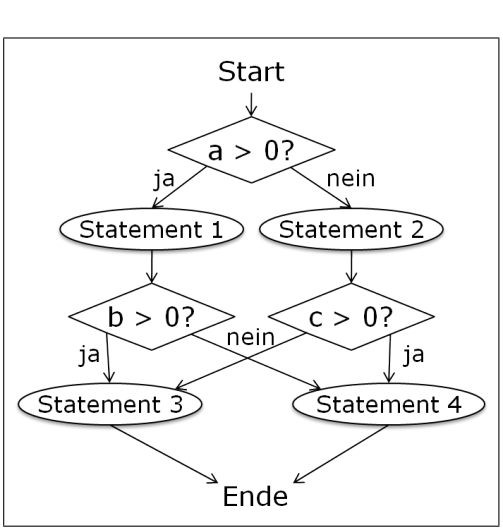
3. Übung – Informatik I für VIW

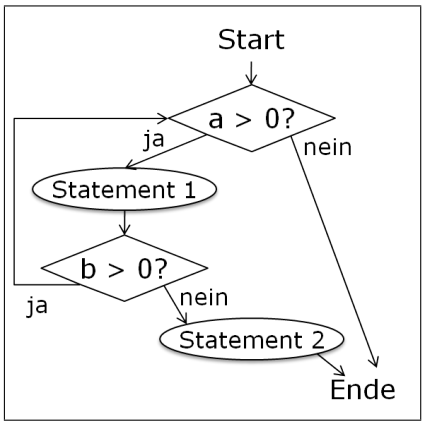
Fakultät Verkehrswissenschaften
Fachrichtung Verkehrsingenieurwesen

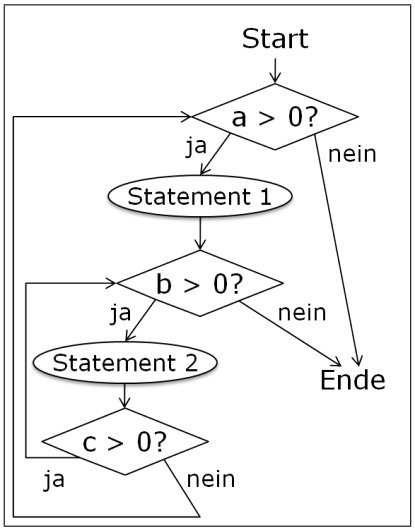
Zeitraum: 15.11. bis 26.11.2010 (WS 2010/11)

Aufgabe 1 – Strukturierte Algorithmen

Ablaufpläne dienen zum graphischen Beschreiben von Algorithmen. Sie können oft nur dann in die Programmiersprache C übertragen werden, wenn sie strukturiert sind. In den folgenden drei Abbildungen sind nicht-strukturierte Ablaufpläne gegeben. Transformieren Sie diese in äquivalente strukturierte Ablaufpläne. Dazu müssen Sie gegebenenfalls Hilfsvariablen einführen und/oder Statements duplizieren.







Aufgabe 2 – Grundstatements von C

Formulieren Sie die drei folgenden Ausschnitte aus C-Programmen nur unter Nutzung des Assignment-, Compound-, If- und While-Statements. Führen Sie gegebenenfalls Hilfsvariablen ein.

```
do                |  for (i = 1; i <= c; i++)    |  do {
{                 |  {                           |     for (; i > 3; i-=d) {
  a++;            |     if (c < d)               |       a -= c;
  if (a < 0)      |       continue;              |       continue;
    break;        |     a++; c--;                |       if (a < d)
  b+=a;           |     if (a > e)               |         break;
}                 |       break;                 |     }
while(a < c);     |  }                           |  } while (a > b);
```

Aufgabe 3 – Programmieren von Schleifen

(a) Welche Ausgabe erzeugt der folgende C-Programm-Ausschnitt?

```
for (i = 1; i < 20; i+=3)
{
  if (i > 5 && i < 10)
    continue;
  if (i % 2 == 0)
    i-=2;
  printf("%d ", i);
  if (30 - i <= i)
    break;
}
```

(b) Schreiben Sie für jede der folgenden Ausgaben ein C-Programm, welches diese Ausgaben erzeugt. In jedem der Programme soll jeweils nur eine `printf`-Anweisung stehen; diese soll bei jedem Aufruf immer nur eine Zahl der Folge ausgeben.

- `2 4 6 8 10 12 9 6 3 0`
- `1 2 3 4 2 3 4 5 3 4 5 6 4 5 6 7`
- `1 4 27 256 3125 46656` ($= n^n$ für $n = 1, \ldots, 6$)

Aufgabe 4 – Binäre Suche

Die binäre Suche ist ein Algorithmus zum Auffinden eines Wertes k in einer sortierten Folge f von Werten. Das Verfahren arbeitet durch schrittweises Halbieren des Suchbereichs und kann in folgende Schritte zerlegt werden:

1. am Anfang ist der Suchbereich die gesamte Folge f,
2. der Wert, der sich in der Mitte des aktuellen Suchbereichs befindet, wird mit k verglichen
3. ist dieser Wert gröser als k, dann wird das Ende des Suchbereichs auf die aktuelle Mitte des Suchbereichs gesetzt
4. ist dieser Wert kleiner als k, dann wird der Anfang des Suchbereichs auf die aktuelle Mitte des Suchbereichs gesetzt
5. solange der Suchbereich mehr als einen Wert beinhaltet, gehe zurück zu Punkt 2
6. der einzige Wert im Suchbereich wird direkt mit k verglichen

Gegeben sind die Variablen `feld` (vom Typ `int`-Array), `laenge` (vom Typ `int`) und `key` (vom Typ `int`). Im Array `feld` befinden sich aufsteigend sortierte Werte in den Einträgen `0` bis `laenge`-1. Schreiben Sie eine Folge von C-Statements, die mit Hilfe der binären Suche bestimmt, ob sich der Eintrag `key` im Array `feld` befindet. Geben Sie das Ergebnis am

Bildschirm aus; wenn sich `key` in `feld` befindet, geben Sie auch die genaue Position in `feld` aus.

Aufgabe 5 (Zusatz) – Langton-Ameise

Gegeben ist ein Raster von quadratischen Feldern, die am Anfang weiß gefärbt sind. Auf einem der Felder befindet sich eine Ameise, die nach Norden schaut und wiederholt die folgenden Aktionen ausführt:

1. Ist das Feld, auf dem die Ameise steht weiß oder schwarz?

   weiß $\rightsquigarrow$ die Ameise dreht sich 90° nach links und färbt das aktuelle Feld schwarz

   schwarz $\rightsquigarrow$ die Ameise dreht sich 90° nach rechts und färbt das aktuelle Feld weiß

2. die Ameise rückt ein Feld in ihrer Blickrichtung nach vorn

Schreiben Sie ein C-Programm, welches die Ameise simuliert (eine Ausgabe — etwa in Form von Text — brauchen Sie nicht zu programmieren). Implementieren Sie das Raster als zweidimensionales Array mit 101×101 Einträgen (dabei kann zum Beispiel der Eintrag 1 für die Farbe weiß und der Eintrag 0 für die Farbe schwarz stehen). Die Ameise soll sich am Anfang in dem mittleren Feld befinden und so lange laufen, bis sie den Rand des Rasters erreicht hat. Bedenken Sie, dass Sie selbst dafür sorgen müssen, dass jedes der Felder am Anfang weiß ist.