Dr. S. Lewandowski **Einführung in die Informatik I (autip)** WS 2007/08

Übungsblatt 06

Ausgabe: 28.11. Abgabeschluss: Mittw., 5.12., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

Von jedem Aufgabenblatt werden maximal 20 Punkte auf den Schein angerechnet.

---

1. (11(+6) Punkte) **Binärzahlen:** Diese Aufgabe soll uns ein tieferes Verständnis der Datentypen im Inneren unseres Rechners vermitteln.

   Im Rechner sind alle Daten durch Vektoren von Bits (mit Wert 0 oder 1) dargestellt. Wir wollen hier nun die Addition von positiven ganzen Zahlen mit 8 Bits simulieren. Mit 8 Bits lassen sich $2^8 = 256$ verschiedene Werte darstellen. Bei der gewöhnlichen Darstellung von Zahlen zur Basis 2 ist der Wert einer Zahl in Binärdarstellung $b_7b_6b_5b_4b_3b_2b_1b_0 = \sum_{i=0}^{7} 2^i \cdot b_i$, so hat 00010110 den Wert $0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 16 + 4 + 2 = 22$.

   Bei der Addition zweier Zahlen geht man wie beim schriftlichen Addieren zweier Dezimalzahlen stellenweise vor (beginnend bei der niedrigstwertigen Stelle). Addieren wir zu 00010110 eine 1, so erhalten wir 00010111. Addieren wir zu dieser Zahl nochmals 1, so ergibt die Addition auf die letzte Stelle einen Überlauf, der auf die Stelle davor übertragen wird. Dort steht jedoch auch eine 1, sodass erneut ein Überlauf passiert usw. Wir erhalten schließlich als Ergebnis 00011000, die Binärdarstellung der Dezimalzahl 24.

   Zur Simulation verwenden wir einen `type SimulNat is array (0..7) of Boolean;` – der Wert an Index `i` sei dabei genau dann `true`, wenn das i-te Bit der simulierten Zahl den Wert 1 hat.

   - (1 Punkt, leicht) Schreiben Sie eine Funktion, die als Eingabe eine 8-Bit-Zahl erhält und den zugehörigen Dezimalwert zurückgibt.
   - (2 Punkte, mittel) Schreiben Sie eine Funktion, die als Eingabe eine Dezimalzahl erhält und die zugehörige 8-Bit-Zahl zurückgibt.

     Hinweis: Ist die Zahl ungerade, so wird das Bit mit der Wertigkeit 2^0 eine 1 sein. Die höherwertigen Stellen erhält man, indem man die Dezimalzahl halbiert und erneut testet, ob diese nun gerade oder ungerade ist. Beispiel: Die Zahl 13 ist ungerade, das Bit 0 muss 1 sein. Ganzzahldivision der 13 durch 2 ergibt 6 – 6 ist gerade, das Bit 1 ist also 0. Ganzzahldivision der 6 durch 2 ergibt 3 – 3 ist ungerade, das Bit 2 ist demnach 1. Ganzzahldivision der 3 durch 2 ergibt 1 – 1 ist ungerade, das Bit 3 ist ebenfalls 1. Ganzzahldivision der 1 durch 2 ergibt 0 – alle höherwertigen Bits sind 0. Als 8-Bit-Zahl erhält man somit die 00001101 – beachten Sie, dass das niedrigstwertige Bit ganz rechts steht (so wie auch bei den Dezimalzahlen die Einer ganz rechts stehen).
   - (2 Punkte, leicht–mittel) Schreiben Sie eine Prozedur `AddEins`, die auf eine gegebene Zahl in Binärdarstellung 1 addiert. Von welcher Art muss der Parameter sein? Wie erkennt man, falls die resultierende Zahl sich nicht mehr mit 8 Bit darstellen lässt?
   - (2 Punkte, mittel) Schreiben Sie eine Funktion `Addiere`, die zwei Binärzahlen mit je 8 Bit als Eingabe erhält und als Ergebnis ebenfalls eine 8-Bit-Zahl zurückgibt. Erkennen Sie auch hier, falls ein Überlauf passiert.
   - Will man negative Zahlen darstellen, so deutet man in der Regel das höchstwertige Bit (hier Bit 7) als -2^7. Dies ist die sogenannte Zweierkomplement-Darstellung. Der mit 8 Bit darstellbare Bereich ist nun -128 bis 127.

     Zu einer positiven 8-Bit-Zahl erhält man die zugehörige negative 8-Bit-Zahl, indem man alle Bits negiert und auf die so erhaltene Zahl 1 mit obigem Algorithmus addiert. Beispiel: Negiert man die 8-Bit-Darstellung von 22 (dies ist 00010110), so erhält man zunächst 11101001, Addition von 1 führt zu 11101010. Diese Zahl hat den Wert $-128 + 64 + 32 + 8 + 2 = -22$.

(2 Punkte, mittel) Schreiben Sie eine Prozedur `Negiere`, die eine 8-Bit-Zahl mit Dezimalwert `x` in eine 8-Bit-Zahl mit Dezimalwert `-x` verwandelt.

(2 Punkte, mittel–schwer) Fügen Sie als Kommentarzeilen mit ein, warum dies funktioniert. Bei welcher Zahl funktioniert das Verfahren nicht?

- **Zusatzaufgabe (6 Punkte, sehr schwer, nur für Tüftler):** Bei der Zweierkomplement-Darstellung gibt jeweils das höchstwertige Bit an, ob eine Zahl negativ ist.

  Eine auf den ersten Blick sehr merkwürdig anmutende Zahldarstellung ist die zur Basis „-2“. Die Binärzahl 1101 hat dann z.B. den Wert $1\cdot(-2)^3+1\cdot(-2)^2+0\cdot(-2)^1+1\cdot(-2)^0 = -8+4+1 = -3$. Die Dezimalzahl +9 hätte die Binärdarstellung 11001 (16-8+1).

  Finden Sie Algorithmen zur Negierung und Addition solcher Zahlen, die zur Basis -2 dargestellt sind.

  Hinweis: Diese Zahldarstellung demonstriert lediglich, dass es Alternativen zur Zweierkomplement-Darstellung gibt. In der Praxis findet sie meines Wissens nach keinerlei Anwendung.

Demonstrieren Sie die von Ihnen implementierten Prozeduren und Funktionen durch geeignete Testfälle im Hauptprogramm.

2. (2+1 Punkte) **`for`-Schleifen**: Wir haben gelernt, dass die Laufvariblen von `for`-Schleifen implizit deklariert werden. So gesehen stellt eine `for`-Schleife ebenfalls einen Block dar.

   (2 Punkte, leicht) Simulieren Sie eine `for`-Schleife durch einen Block und eine `while`-Schleife. Testen Sie Ihr Vorgehen an dem Beispiel:

```
for i in 1..10 loop
  for i in 1..10 loop
    put(i,0);
  end loop;
end loop;
```

   (1 Punkt, leicht) Fügen Sie als Kommentarzeilen neben Ihrer Idee hinzu, ob und (wenn ja) welche Eigenschaften der Laufvariablen von der Simulation nicht abgebildet werden.

3. (5+1 Punkte) **Minimale Differenz:** Gegeben seien 6 paarweise verschiedene Ziffern zwischen 0 und 9. Diese sind so in zwei 3-stellige Zahlen aufzuteilen, dass die Differenz zwischen diesen beiden Zahlen möglichst gering ist. Beispiele: aus den Ziffern 1, 3, 4, 6, 8, 9 lassen sich die Zahlen 398 und 416 bilden, die minimale Differenz ist 18; aus den Ziffern 0, 1, 2, 4, 5, 6 lassen sich die Zahlen 165 und 204 bilden, die minimale Differenz ist 39 (alternative Lösung: 462 und 501, es reicht in solchen Fällen 1 Lösung anzugeben).

   (a) (5 Punkte, mittel–schwer) Schreiben Sie ein Ada-95-Programm, das 6 Zahlen einliest (diese müssen nicht in aufsteigender Reihenfolge eingegeben werden), aus diesen dann zwei dreistellige Zahlen bildet, deren Differenz möglichst gering ist, und diese Zahlen sowie die Differenz ausgibt.

   Gliedern Sie das Hauptprogramm dabei in eine Prozedur `Get(feld)`, die eine Variable vom Typ eines 6-elementigen Feldes einliest, ein `Berechne2Zahlen(...)`, das aus diesem Feld die zwei Zahlen berechnet, und eine `Ausgabe(...)`, die diese Zahlen und deren Differenz ausgibt. Überlegen Sie und begründen Sie (als Kommentarzeilen einfügen), welche Parameterübergabemechanismen hier jeweils verwendet werden sollten.

   Hinweis: Wer sich nicht an diese vorgegebene Struktur im Hauptprogramm hält (und alles in eine Prozedur zusammenfasst) muss mit Punktabzug rechnen.

   (b) (1 Punkt, leicht–mittel) Überlegen und erläutern Sie (bitte als Kommentarzeilen einfügen), ob (und wenn ja wie) Sie das Programm ändern müssten, wenn auch gleiche Zahlen in der Eingabe erlaubt sind (falls Änderungen nötig sind, müssen diese nicht ausprogrammiert werden).
