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

Übungsblatt 05

Ausgabe: 21.11. Abgabeschluss: Mittw., 28.11., 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. (3+4 Punkte) **Primzahlen:** Ein sehr einfaches Primzahltestprogramm überprüft, ob die eingegebene Zahl $n \geq 2$ sich ganzzahlig durch eine Zahl aus dem Intervall von 2 bis $n-1$ teilen lässt. Ist dies für keine Zahl i mit $2 \leq i \leq n-1$ der Fall, d.h. n hat außer der 1 und sich selbst keine Teiler, so ist n eine Primzahl. (Anmerkung: Primzahlen haben immer genau 2 Teiler: die 1 und sich selbst; d.h., die 0 und 1 sind keine Primzahlen)

   (a) (3 Punkte, leicht–mittel) Alle ganzen Zahlen $n \geq 2$ lassen sich eindeutig als Produkt von Primzahlen schreiben, der sogenannten Primfaktorzerlegung. Schreiben Sie eine Prozedur mit Parameter n, die die Primfaktorzerlegung von n ausgibt. Benutzen Sie Ihre Prozedur, um im Hauptprogramm die Primfaktorzerlegungen von 15, 24, 108, 153 und 1001 auszugeben.

   (b) (4 Punkte, mittel) Will man alle Primzahlen bis zu einer Zahl n bestimmen, so kann man diese mit dem Sieb des Eratosthenes berechnen. Zunächst schreibt man sich dazu alle Zahlen von 2 bis n auf. Die 2 ist Primzahl und man markiert alle echten Vielfachen der 2 (diese lassen sich ganzzahlig durch 2 teilen und sind somit keine Primzahlen). Die nächste unmarkierte Zahl, die 3, ist Primzahl und man markiert wieder alle echten Vielfachen dieser Zahl. Dieses wird jeweils mit der nächsten unmarkierten Zahl wiederholt.

   Die unmarkierten Zahlen sind nicht Vielfache einer kleineren Zahl, sie sind also Primzahlen. Wir erhalten so der Reihe nach 2, 3, 5, 7, 11, ...

   Schreiben Sie eine Prozedur `Primzahlsieb` mit Parameter n, die das Sieb das Eratosthenes durchführt und dann alle Primzahlen bis zur Zahl n ausgibt. (Hinweis: Sie können Parameter einer Prozedur als Feldgrenzen verwenden)

   (c) **Zusatzaufgabe (schwer, 3 Punkte):** Überlegen Sie, ob man das Sieb des Eratosthenes benutzen kann, um die Berechnung der Primfaktorzerlegung zu beschleunigen. Welchen Aufwand hat Ihr Algorithmus aus Teil (b), mit welchem Aufwand müsste man bei Verwendung des Siebs des Eratosthenes rechnen? Skizzieren Sie Ihre Überlegungen auch an einigen Beispielen.

2. (5+2 Punkte) **Hotel Devil's Door:** In diesem merkwürdigen Hotel sind die Zimmer mit den Zahlen 1 bis 100 nummeriert. Nachts, wenn alle schlafen, rennen 100 kleine Teufel durch die Gänge und schlagen die Türen auf und zu.

   - Der erste Teufel öffnet alle Türen.
   - Der zweite Teufel schließt jede zweite Tür. Das Zimmer 1 ist also noch offen, das Zimmer 2 geschlossen, Zimmer 3 offen, usw.
   - Der dritte Teufel verändert den Zustand jeder dritten Tür: Ist die Tür offen, so schließt er sie – ist sie geschlossen, so öffnet er sie. Er schließt also Tür 3, öffnet Tür 6, schließt Tür 9, usw.
   - Der i-te Teufel verändert den Zustand jeder i-ten Tür – analog zum dritten Teufel.

   Wir wollen nun das nächtliche Treiben in diesem seltsamen Hotel simulieren.

(a) (5 Punkte, mittel) Schreiben Sie ein Programm, das das nächtliche Türen-auf-und-zuschlagen im Hotel Devil's Door simuliert. Geben Sie jeweils den Zustand der Türen aus, nachdem ein weiterer Teufel durch die Gänge gelaufen ist.

(b) (2 Punkte, mittel–schwer) In welchen Zimmern kann man schlafen, so dass am nächsten Morgen die Zimmertür geschlossen ist? In welchen Zimmern, deren Tür morgens geschlossen ist, wurde die Tür am seltensten auf und zu gemacht? Begründen Sie Ihre Antwort.

(c) **Zusatzaufgabe (mittel–schwer, 2 Punkte):** In welchen Zimmern hört man vom Türenklappern am wenigsten, in welchen am meisten (solange die eigene Tür offen ist, hört man stets, wenn eine anderen Tür geöffnet oder geschlossen wird)? Ermitteln Sie die Lösung durch ein Programm.

3. (2+4 Punkte) **Goto Exit:** Die Verwendung der beiden Ada-Befehle `goto` (unbedingter Sprung zu einem Label, wobei (in Ada) ein Sprung von außen in den Rumpf einer Schleife verboten ist) und `exit` (verlassen der umliegenden Schleife) führen schnell zu unverständlichen und nicht mehr nachvollziehbaren Programmen und sollten daher grundsätzlich vermieden werden. Untersuchen Sie zur Abschreckung folgende Funktion:

```
function Nixcapito(Arg:Positive) return Positive is
   X:Positive:=Arg;
   Z:Positive:=1;
   T:Positive:=Z;
begin
   loop
      if Z mod T /= 0 then
         goto Lab2;
      end if;
      <<Lab1>>
      T:=1; Z:=Z+1;
      <<Lab2>>
      T:=T+1;
      if Z<T*T then
         if X=1 then
            exit;
         else
            X:=X-1; goto Lab1;
         end if;
      end if;
   end loop;
   return Z;
end;
```

(2 Punkte, mittel) Was berechnet diese Funktion? Beschreiben Sie das Vorgehen des Programms. (Fügen Sie beides als Kommentarzeilen zu Beginn des Programms ein.)

(4 Punkte, mittel–schwer) Schreiben Sie nun diese Funktion so um, dass sie ohne `goto` und `exit` auskommt. Dabei soll die Folge der Einzelschritte (bzw. die Idee des obigen unstrukturierten Algorithmus) soweit möglich beibehalten werden.
